

APPENDIX B—METHODS

This appendix contains the procedures, descriptions of computer codes, and input/output files used for the widespread fatigue damage (WFD) analyses as discussed in section 5. The complete UNIX shell scripts, FORTRAN source codes, and the data used for the current WFD programs are provided on a separate CD.

Page No.	Description
B-2	Equivalent initial flaw size analysis procedures and shell script file
B-7	Computer code get_eifs.f for collapsing measured EIFS
B-13	Computer code betauk.f for determining analytical-experimental correction factor
B-16	Computer code efsiz.f for determining EIFS sizes
B-20	Finite element global-local mapping procedures
B-23	Computer code fmstrs_nl.f for extracting global NASTRAN nonlinear solutions
B-26	Computer code genfeam_nl.f for mapping global solutions to the local model
B-31	Computer code meshfem.f for converting ABAQUS files to TKALT for the local model
B-34	Program input instruction for running EPFEAM code TKALT
B-59	Computer program for running NASTRAN-to-STAGS conversion
B-71	A brief description of some of the files for 2DFEAM code

B.1 COMPUTATION OF AVERAGE EIFS SIZES.

The equivalent initial flaw size (EIFS) analysis steps are coded in the UNIX ksh script betau for each crack location of all EIFS test panels that were examined using scanning electronic microscope. The analysis steps are outlined as follows:

- Prepare the experimentally measured crack growth data and store them in a directory, denoted as 1 in the figure B-1. The format of the data sets is shown in table B-3.
- Running computer code get_eifs to synchronize the measured data into a small band and to fit a polynomial curve through the synchronized crack growth history, denoted as 2 in figure B-1.
- Using the baseline FASTRAN-II input file, denoted as 4 in figure B-1, and the modified crack growth history, denoted as 3 in figure B-1, to determine the correction factor (β_u) using the computer code betauk, denoted as 5 in figure B-1. The processes are iterated six times to obtain good convergence of β_u factors.
- Once the final β_u is determined, the baseline FSTRAN input file is modified with the β_u factor, denoted as 6 in figure B-1.
- The next step is to run FASTRAN-II code using modified FASTRAN-II input files. The FASTRAN-II analyses are performed using 16 various EIFS sizes. Store crack growth history to the crack growth matrices, denoted as 7 in figure B-1.
- The EIFS sizes can be interpolated for each measured data point from the crack growth matrices. The average EIFS sizes are calculated using the computer code efsizes, denoted as 8 in figure B-1. The outputs are stored in a file eifise.out, denoted as 9 in figure B-1.

The script file and the source codes for get_eifs, betauk, efsizes are discussed in the following appendices. The input and output data for all EIFS data points are provided on the CD.

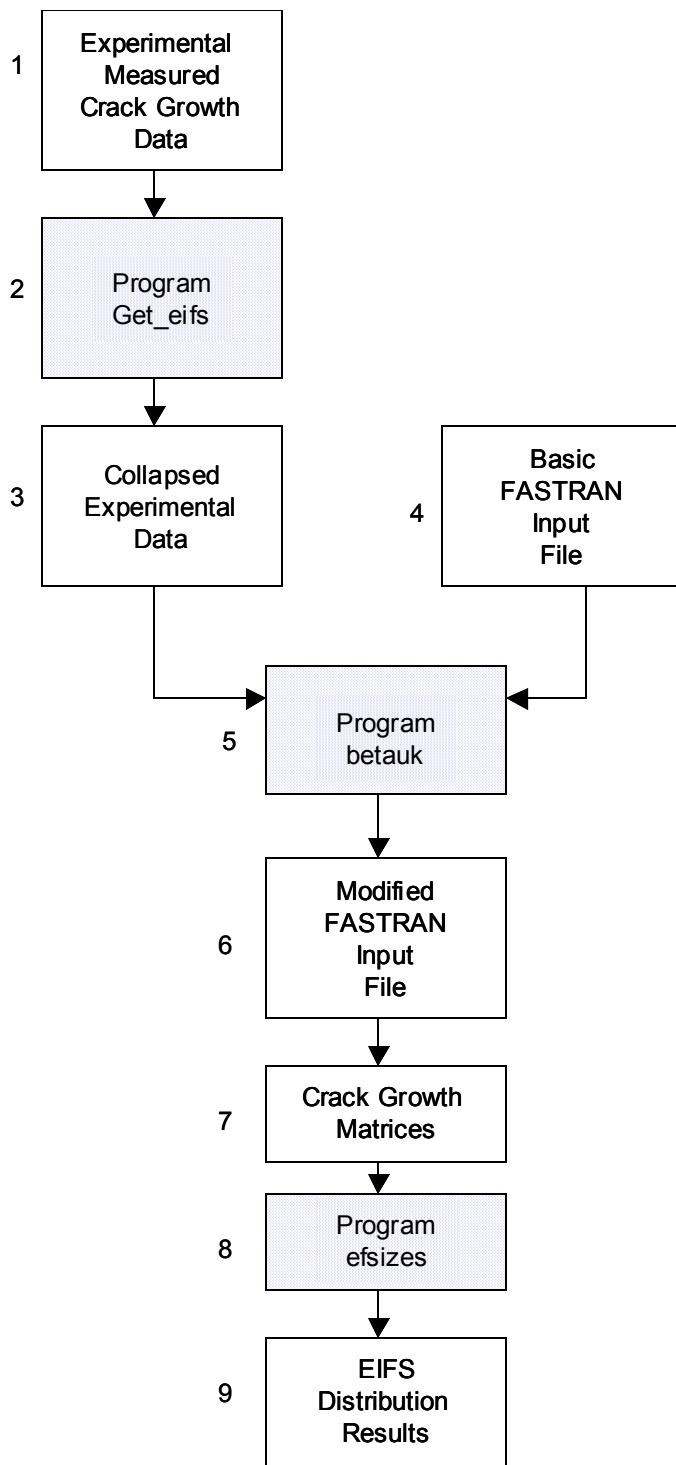


FIGURE B-1. FLOWCHART OF EIFS ANALYSIS, CODED IN THE SHELL SCRIPT

TABLE B-1. ksh SCRIPT betau FOR RUNNING EIFS ANALYSIS

```

#
=====
#      "E I F S   Analysis"
#      Ching Hsu   Boeing, Long Beach
#
=====
clear
banner "EIFS"
#   use dof=2 for all except panel no. 7 (use 4)
echo " Enter degree of polynominal equation \n      (use 4 for EIFS-7 and 2 for
others) ..\c"; read dof
echo "$dof">>ndeg.dat
#ls -1 get_eifs.in_*
echo " Enter ID ... \c";read id
runid=$id
echo "$runid">>efscntl
count=1;last=6

# --- change to EIFS directory
cd ~/WFD/EIFS/Fatigue/Type_2/Beta45

# --- get crack growth test data and collapse the experimental data
ln -sf get_eifs.in_$runid get_eifs.in
get_eifs<get_eifs.in
ln -sf mod_cyc.dat_$runid mod_cyc.dat

# --- setup FASTRAN input files
cp fast2.in_betau_$runid fast2.betau.in
echo "fast2.betau.in" > tempfile_fast

# --- change initial crack sizes in the input file
chngac < chngac.in

# --- reset beta factor modification file
rm mod1.dat

# --- begin of iteration
while [ "$count" -le $last ]
do
  fast2.2.exe < tempfile_fast
  echo " read fastrans2 ...";run read_fast2 x
  echo " runnig betauk ...$count "; betauk<betauk.in

# --- change the initial flaw
  if [ "$count" -eq "1" ]
    then
      echo " running chngac ...";chngac<chngac.in_betauk
    fi
    let count=$count+1
done

cp fast2.betau.in fast2.in_betau_"$runid"_fnl
cp get_eifs.out_ftd get_eifs.out_ftd_$runid

```

```

#
=====
#    computes EIFS
#
=====
clear
# get id of the crack growth database
ver='1'
ver=''
ls -1 *fnl$ver
while read x;do runid=$x;done<efs.cntl
# echo " enter run id ...\";read runid
echo " run id =$runid "

count=1;last=16

# --- clean up files
rm fast2_crkgro.dat_$runid

# --- set up input file for fastran
echo "fast2.betau.in " > tempfile_fast

# --- set up initial crack sizes
genefs <genefs.in

# --- create generic fastan input file
cp fast2.in_betau_"$runid"_fnl$ver fast2.betau.in

# --- execute fastran
while [ "$count" -le $last ]
do

# --- update fastran input file: fast2.$runid.in
chnefs <chnefs.in

# --- run fastran
echo " running fast2 "
fast2.2.exe < tempfile_fast

# --- extracting crack growth history
read_fast2 < read_fast2.in

# --- save analytical growth history to a file
cat fast2_crkgro.dat >> fast2_crkgro.dat_$runid

# --- repeat
let count=$count+1
done

```

```
# --- copy all prediction to a generic file for plotting
cp fast2_crkgro.dat_${runid} fast2_crkgro.dat
cp fast2_crkgro.dat_${runid} fast2_crkgro.dat_${runid$ver}
efsiz < efsiz.in

cp crack_vs_cyc.dat crack_vs_cyc.dat_${runid$ver}
cp efsiz.out efsiz.out_${runid$ver}
cp efsiz.log efsiz.log_${runid$ver}

echo " ===== E N D ===== "
```

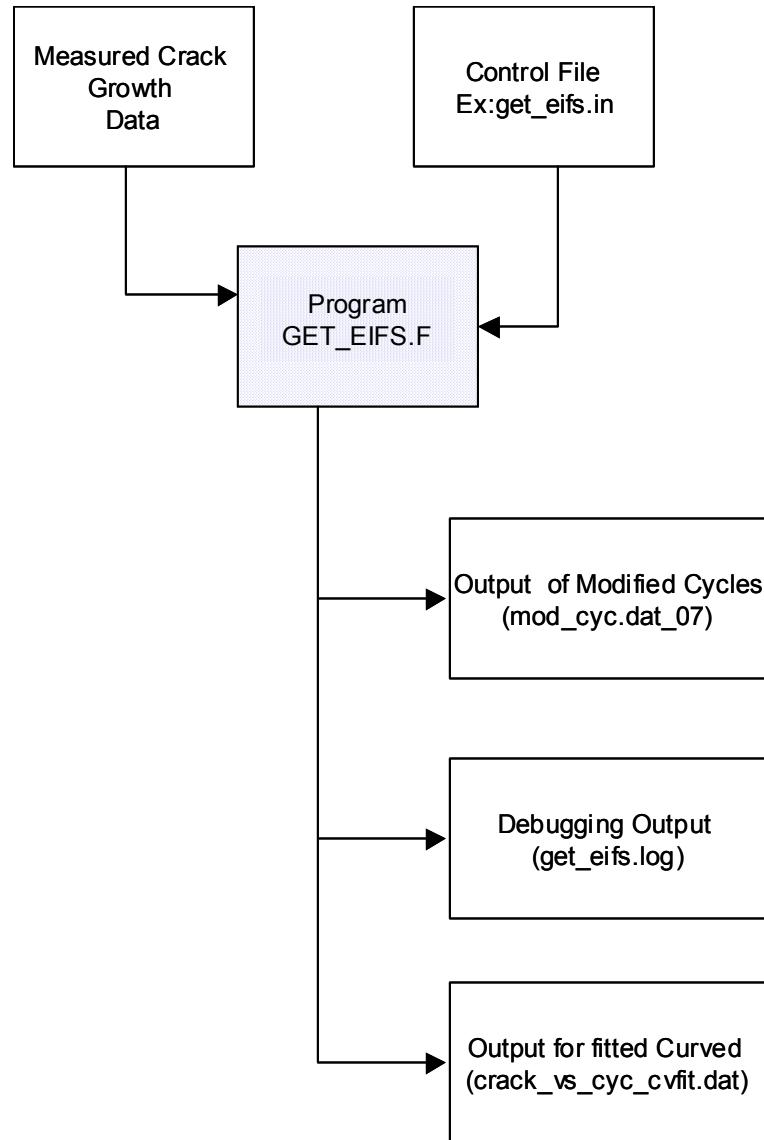


FIGURE B-2. INPUT AND OUTPUT FILES FOR PROGRAM *get_eifs.f*

TABLE B-2. TYPICAL INPUT FILE FOR PROGRAM get_eifs.f

FILENAME get_eifs.in

/uhs1112013/d1xr/c045270/WFD/EIFS/Fatigue/Type_2/Beta45/SEM/

mod_cyc.dat_07

sem_07A06L.dat

sem_07A06R.dat

sem_07A07L.dat

sem_07A07R.dat

sem_07A08L.dat

sem_07A12R.dat

sem_07A13R.dat

sem_07A14L.dat

sem_07A14R.dat

sem_07A15R.dat

sem_07A16L.dat

sem_07A16R.dat

sem_07A17L.dat

sem_07A17R.dat

Directory for the measured
crack growth datasets

Filename of the modified crack
growth history

Filenames of measured crack
growth history

TABLE B-3. TYPICAL INPUT FILE FOR PROGRAM get_eifs.f—MEASURED CRACK GROWTH HISTORY

FILENAME sem_07a07l.dat

```
# /uhs1112013/d1xr/c045270/WFD/EIFS/Test_data/SEM/sem_07A07L.dat
#   28    07A07L
 1  93840  .042  .011
 2  95710  .043  .009
 3  99010  .046  .007
 4  100880  .048  .003
 5  104180  .052  .011
 6  106050  .054  .014
 7  107810  .056  .015
 8  111220  .059  .016
 9  112980  .060  .019
10  114520  .062  .020
11  116390  .063  .019
12  119690  .065  .020
13  121560  .067  .012
14  123320  .067  .015
15  124860  .069  .015
16  126730  .070  .017
17  128490  .072  .018
18  130030  .072  .015
19  131900  .073  .016
20  133660  .075  .015
21  135200  .077  .011
22  137070  .079  .013
23  138830  .081  .013
24  140370  .083  .015
25  142240  .085  .015
26  144000  .087  .016
27  145540  .089  .015
28  147410  .092  .015
```

Filename

Crack ID

Number of data points

Data point number

Crack length and depth

TABLE B-4. TYPICAL OUTPUT FILE FOR PROGRAM get_eifs.f

FILENAME mod_cyc.dat_07

```
# Date: 22-Oct-01 Time: 11:28:44
```

```
# sem_07A06L.dat Mod Cyc= 26438
# 38
 80200    53668 .0260
 82070    55538 .0290
 85370    58838 .0330
 87240    60708 .0340
 89000    62468 .0350
 90540    64008 .0360
 92410    65878 .0370
 95710    69178 .0390
 97580    71048 .0400
 99340    72808 .0430
100880    74348 .0460
102750    76218 .0480
104510    7978 .0510
106050    79518 .0540
107920    81388 .0570
. . . . .
. . . . .
131900   105368 .0880
133770   107238 .0910
135530   108998 .0940
137070   110538 .0970
138940   112408 .1010
140700   114168 .1040
142240   115708 .1080
144110   117578 .1150
145870   119338 .1180
147410   120878 .1230
```

Modification cycles for the crack

Crack length

Modified number of cycles

Original number of cycles

```
# sem_07A06R.dat Mod Cyc= 19200
```

```
# 60
 28500    9206 .0040
 30260   10966 .0060
. . . . .
. . . . .
. . . . .
```

TABLE B-5. TYPICAL OUTPUT FILE FOR PROGRAM get_eifs.f

FILENAME get_eifs.log

1	.000332	26438	38	sem_07A06L.dat
2	.000505	19200	60	sem_07A06R.dat
3	.000000	30412	28	sem_07A07L.dat
4	.000000	54134	36	sem_07A07R.dat
5	.000000	96503	10	sem_07A08L.dat
6	.000040	33201	50	sem_07A12R.dat
7	.000006	46681	42	sem_07A13R.dat
8	.000772	12576	38	sem_07A14L.dat
9	.000052	26338	55	sem_07A14R.dat
10	.000288	35948	46	sem_07A15R.dat
11	.014971	26730	31	sem_07A16L.dat

File name for the original Crack growth history

Number of data points

Modification cycles

Fitted initial crack length

crack number

TABLE B-6. TYPICAL OUTPUT FILE FOR PROGRAM get_eifs.f—FITTED CURVE

FILENAME crack_vs_cyc_cvfit.dat

Initial crack length= .0038380

#no	Cycle	Length	Da/Dn	a/R
1	1.00000E+00	3.83799E-03	2.43867E-07	4.09386E-02
2	3.70635E+03	4.82088E-03	2.87223E-07	5.14227E-02
3	7.41170E+03	5.96914E-03	3.33011E-07	6.36709E-02
4	1.11171E+04	7.29016E-03	3.80321E-07	7.77617E-02
5	1.48224E+04	8.78771E-03	4.28139E-07	9.37356E-02
6	1.85278E+04	1.04617E-02	4.75402E-07	1.11592E-01
7	2.22331E+04	1.23082E-02	5.21061E-07	1.31287E-01
8	2.59384E+04	1.43193E-02	5.64149E-07	1.52739E-01
9	2.96438E+04	1.64839E-02	6.03838E-07	1.75829E-01
10	3.33491E+04	1.87883E-02	6.39492E-07	2.00409E-01
11	3.70545E+04	2.12167E-02	6.70704E-07	2.26311E-01
12	4.07599E+04	2.37521E-02	6.97322E-07	2.53356E-01
13	4.44652E+04	2.63778E-02	7.19461E-07	2.81364E-01
14	4.81706E+04	2.90778E-02	7.37496E-07	3.10163E-01
15	5.18759E+04	3.18379E-02	7.52050E-07	3.39604E-01
16	5.55813E+04	3.46468E-02	7.63972E-07	3.69565E-01
17	5.92866E+04	3.74965E-02	7.74307E-07	3.99963E-01
18	6.29920E+04	4.03834E-02	7.84272E-07	4.30756E-01
19	6.66973E+04	4.33087E-02	7.95230E-07	4.61959E-01
20	7.04027E+04	4.62786E-02	8.08668E-07	4.93639E-01
21	7.41080E+04	4.93054E-02	8.26192E-07	5.25925E-01
22	7.78134E+04	5.24074E-02	8.49525E-07	5.59012E-01
.
.
.
.
37	1.33394E+05	1.51926E-01	4.32967E-06	1.62055E+00
38	1.37099E+05	1.69519E-01	5.20145E-06	1.80821E+00
39	1.40804E+05	1.90747E-01	6.30216E-06	2.03464E+00
40	1.44510E+05	2.16584E-01	7.70296E-06	2.31023E+00

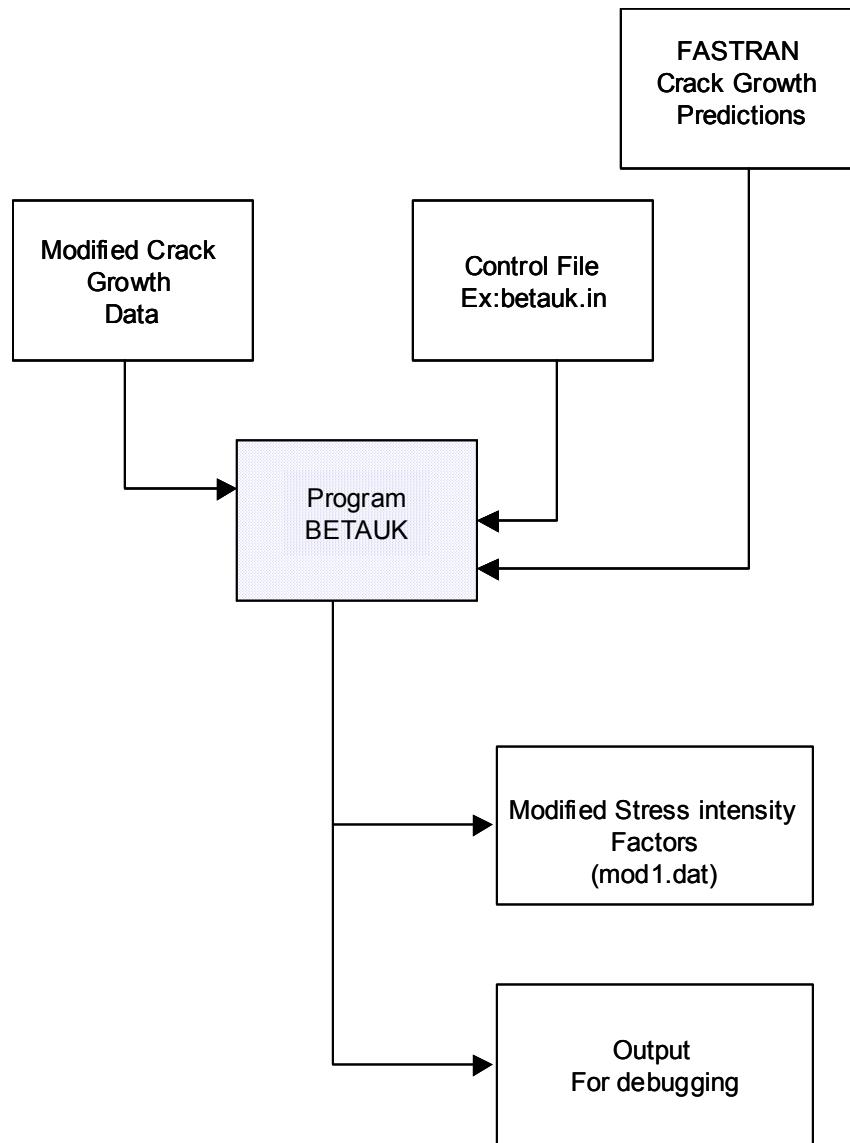


FIGURE B-3. INPUT AND OUTPUT FILES FOR PROGRAM betauk.f

TABLE B-7. TYPICAL INPUT FILE FOR PROGRAM betauk.f

FILENAME betauk.in

/uhs1112013/d1xr/c045270/WFD/EIFS/Fatigue/Type_2/Beta45/

fast2.betau.in

fast2_crkgro.dat

mod_cyc.dat

4.08 1.0

0 0

mod1.dat

File Directory

Filename for FASTRAN input data

FASTRAN prediction

Modified Crack growth History
(see table B-4)

Paris' coefficients 'n' in the equation of
 $da/dN = c\Delta K^n$

Modification cycles

TABLE B-8. TYPICAL OUTPUT FILE FOR PROGRAM bettau.f

FILENAME mod1.dat

#no	Length	T-Beta	N-beta	Last B	Next Ho	a/R
1	4.79677E-03	8.11503E-01	9.20139E-01	8.81936E-01	1.00000E+00	5.11656E-02
2	5.29677E-03	8.00263E-01	9.18846E-01	8.70943E-01	1.00000E+00	5.64989E-02
3	6.79677E-03	7.71304E-01	9.18094E-01	8.40114E-01	1.00000E+00	7.24989E-02
4	9.29677E-03	7.31076E-01	9.16667E-01	7.97537E-01	1.00000E+00	9.91656E-02
5	1.27968E-02	6.85757E-01	9.15623E-01	7.48951E-01	1.00000E+00	1.36499E-01
6	1.72968E-02	6.37677E-01	9.14933E-01	6.96966E-01	1.00000E+00	1.84499E-01
7	2.27968E-02	5.88560E-01	9.15956E-01	6.42564E-01	1.00000E+00	2.43166E-01
8	2.92968E-02	5.40852E-01	9.20472E-01	5.87581E-01	1.00000E+00	3.12499E-01
9	3.67968E-02	4.98927E-01	9.33133E-01	5.34680E-01	1.00000E+00	3.92499E-01
10	4.52968E-02	4.72682E-01	9.63695E-01	4.90489E-01	1.00000E+00	4.83166E-01
11	5.47968E-02	4.76013E-01	1.02632E+00	4.63808E-01	1.00000E+00	5.83166E-01
12	6.52968E-02	5.06424E-01	1.09595E+00	4.62088E-01	1.00000E+00	6.83166E-01
13	7.67968E-02	5.18149E-01	1.07277E+00	4.83000E-01	1.00000E+00	7.83166E-01
14	8.92958E-02	5.26947E-01	1.02154E+00	5.15835E-01	1.00000E+00	8.83166E-01
15	1.02797E-01	5.45785E-01	9.88681E-01	5.52033E-01	1.00000E+00	1.09650E+00
16	1.17297E-01	5.68731E-01	9.69307E-01	5.86739E-01	1.00000E+00	1.25117E+00
17	1.32797E-01	5.93937E-01	9.58274E-01	6.19799E-01	1.00000E+00	1.41650E+00
18	1.49297E-01	6.18804E-01	9.51107E-01	6.50615E-01	1.00000E+00	1.59250E+00
19	1.66797E-01	6.41300E-01	9.45127E-01	6.78533E-01	1.00000E+00	1.7E+00
20	1.85297E-01	6.61099E-01	9.42794E-01	7.01213E-01	1.00000E+00	1.8E+00
21	2.04797E-01	6.80951E-01	9.38806E-01	7.25337E-01	1.00000E+00	1.9E+00
22	2.25297E-01	6.99142E-01	9.38567E-01	7.44903E-01	1.00000E+00	2.40317E+00
23	2.46797E-01	7.16276E-01	9.38567E-01	7.64903E-01	1.00000E+00	2.63250E+00
24	2.69297E-01	7.30951E-01	9.3	7.84903E-01	1.00000E+00	2.87250E+00
25	2.92797E-01	7.3	7.84903E-01	8.13689E-01	1.00000E+00	3.12317E+00
26	3.17297E-01	7.3	7.84903E-01	8.28186E-01	1.00000E+00	3.38450E+00
27	3.42797E-01	7.3	7.84903E-01	8.33142E-01	1.00000E+00	3.65650E+00
28	3.69297E-01	7.84510E-01	9.33117E-01	8.40719E-01	1.00000E+00	3.93917E+00
29	3.96797E-01	7.95798E-01	9.33092E-01	8.52838E-01	1.00000E+00	4.23250E+00
30	4.25297E-01	8.07641E-01	9.33092E-01	8.65553E-01	1.00000E+00	4.53650E+00

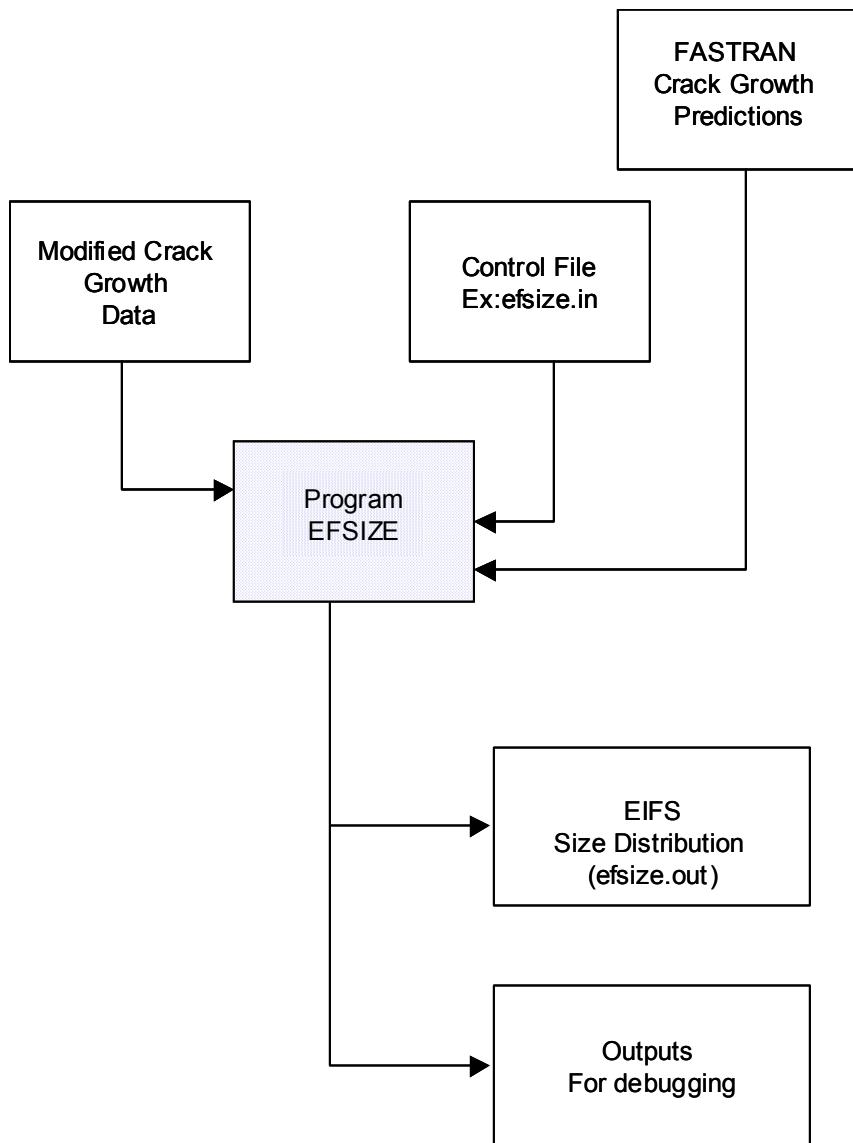


FIGURE B-4. INPUT AND OUTPUT FILES FOR PROGRAM efsize.f

TABLE B-9. TYPICAL INPUT FILE FOR PROGRAM efsize.f

FILENAME efsize.in

fast2_crkgro.dat

mod_cyc.dat

Predicted Crack growth history for
various EIFS

Modified Crack growth History

TABLE B-10. TYPICAL OUTPUT FILE FOR PROGRAM efsize.f

FILENAME efsize.out

#	seq	NPT	EIFS	Crack
1	38	6.696E-04	07A06L	
2	60	8.902E-04	07A06R	
3	28	5.676E-04	07A07L	
4	36	3.056E-04	07A07R	
5	10	1.598E-04	07A08L	
6	50	5.331E-04	07A12R	
7	42	3.663E-04	07A13R	
8	38	1.174E-03	07A14L	
9	55	6.669E-04	07A14R	
10	46	5.034E-04	07A15R	
11	31	6.422E-04	07A16L	
12	19	1.386E-04	07A16R	
13	39	4.079E-04	07A17L	
14	57	9.992E-04	07A17R	

Number of crack= 14 Point=549 Average EIFS= .00057

Average EIFS size for the specimen

Crack ID

Average EIFS for the crack

Number of data points

TABLE B-11. TYPICAL DEBUGGING FILE FOR PROGRAM efsize.f

FILENAME efsize.log

```
# 1 No. of pts= 38 File ->>sem_07A06L.dat
```

#	Seq	Cycles	Length	EIFS
	1	80200	.02600	5.00194E-04
	2	82070	.02900	5.23457E-04
	3	85370	.03300	5.36944E-04
	4	87240	.03400	5.23677E-04
	5	89000	.03500	5.09950E-04
.
.
.
.
36	144110	.11500	8.97948E-04	
37	145870	.11800	8.72921E-04	
38	147410	.12300	8.77907E-04	

Crack ID

```
# 2 No. of pts= 60 File ->>sem_07A06R.dat
```

#	Seq	Cycles	Length	EIFS
	1	28500	.00400	4.98639E-04
	2	30260	.00600	6.69808E-04
	3	31800	.00600	6.35195E-04
.
.
.
.
.
.

Corresponding EIFS for
the measured data point

B.2 FINITE ELEMENT MODEL GLOBAL-LOCAL MAPPING.

The analysis steps are coded in the UNIX ksh script file, mapp, for mapping the boundary conditions from global NASTRAN model to local EPFEAM model. The analysis steps are outlined as follows:

- Prepare the global models with lead cracks, denoted 1 in figure B-5. Set the maximum load to 20 ksi for MSD flat panel models and the analytical load increment at 10% of the maximum load.
- Submit the global models to NASTRAN using nonlinear solution, sol 106. Save the ASCII outputs which contain the displacements of grid points, stress of quad4 and tria3, and forces of bar elements, denoted as 2 in figure B-5.
- Running the computer program fmstrs_nl to extract the pertinent displacements, stresses and loads, denoted as 4 in figure B-5. The extracted boundary conditions will be used for the local model, denoted as 3 in figure B-5. The main purpose of this intermediate step is to provide the listed outputs for error checking.
- The outputs of the fmstrs_nl, denoted as 5 in figure B-5, are used as the inputs for the mapping program genfeam_nl, denoted as 6 in figure B-5. The program will create complete input decks for the EPFEAM code tkalt, denoted as 7 in figure B-5.

The script file, mapp are listed as table B-12. The input and output data for all MSD flat panels are provided in the electronic media.

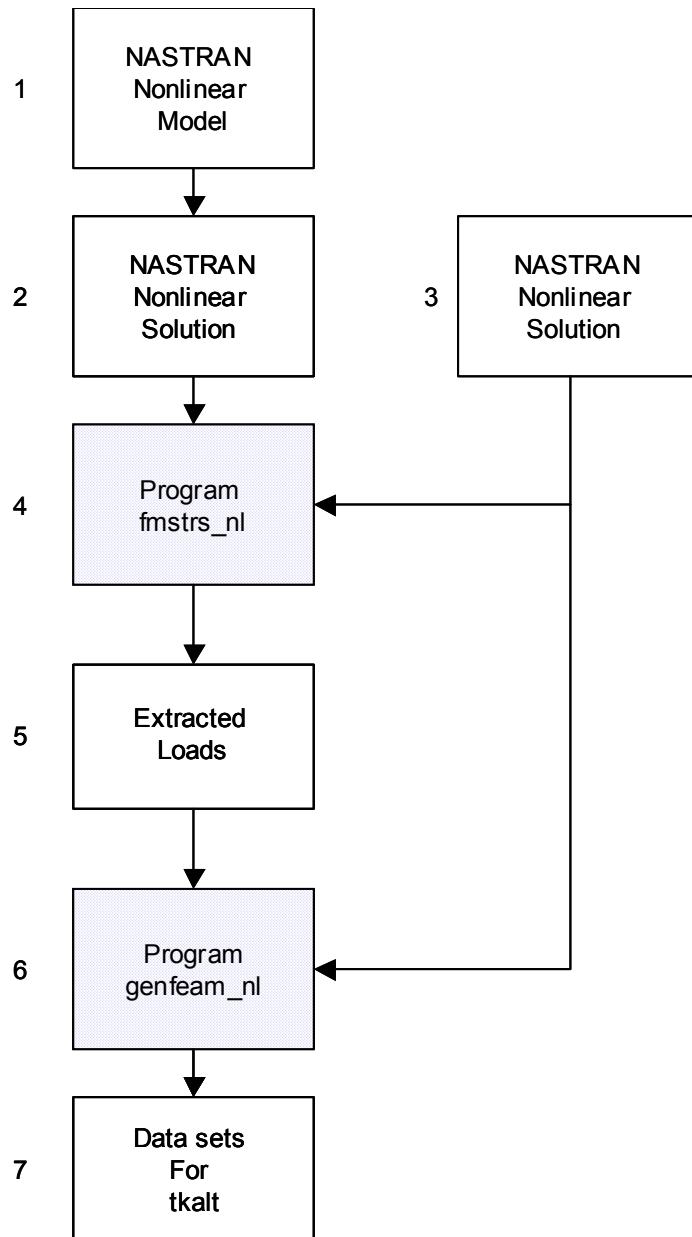


FIGURE B-5. FLOWCHART OF MAPPING GLOBAL-LOCAL BOUNDARY CONDITIONS

TABLE B-12. ksh SCRIPT FOR RUNNING GLOBAL-LOCAL MAPPING ANALYSIS

```

# =====
# generate local boundary conditions
# Ching Hsu
# =====
clear
banner " MAPPING"
echo " Enter first and last run number ....\c" ; read run_1 run_2

##cp /uhs1112013/d1xr/c045270/WFD/MSD/Type_2/fmstrs_step.dat load_step.in
count=$run_1;last=$run_2

while [ "$count" -le "$last" ]
do
    echo " Running load extraction for" $count
    cp fmstrs.in_$count fmstrs.in
    fmstrs_nl <fmstrs.in

    echo " Running mapping for" $count
    run_gen=1
    while [ "$run_gen" -le 20 ]
    do
        ss="$count"_"$run_gen"
        echo " cp genfeam.in_$ss genfeam.in_$run_gen"
        genfeam_nl < genfeam.in_$run_gen
        let run_gen=$run_gen+1
    done
    let count=$count+1
done

# ----- E N D -----

```

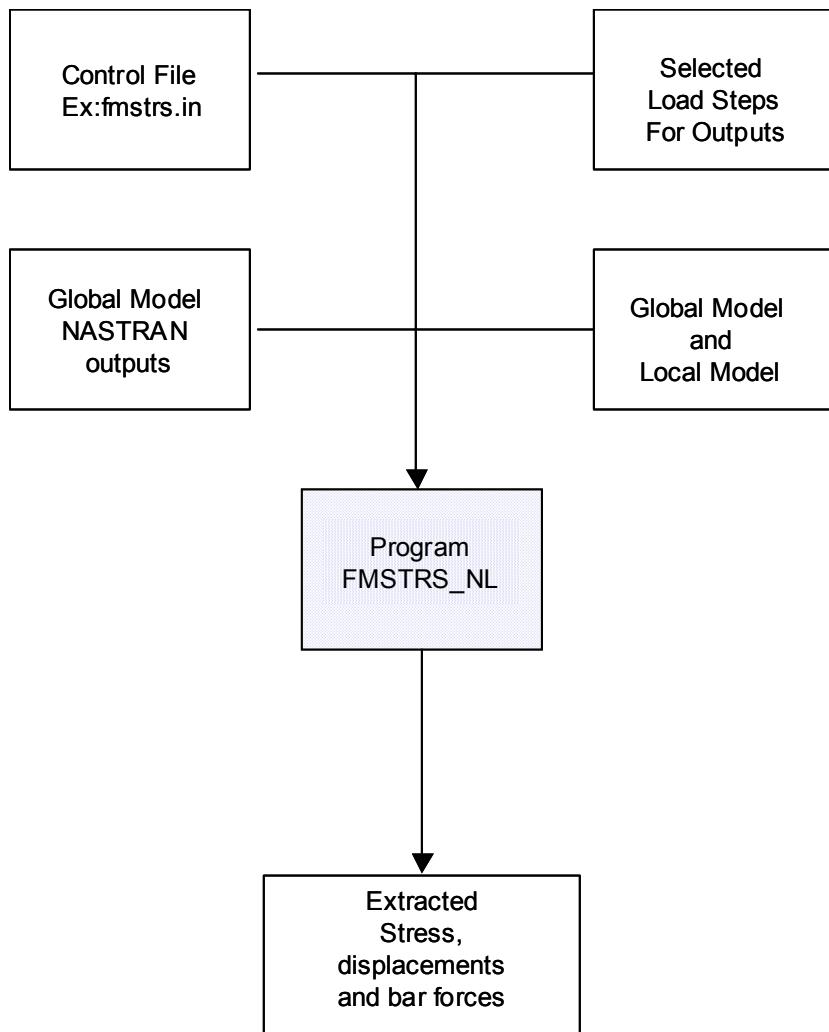


FIGURE B-6. INPUT AND OUTPUT FILES FOR PROGRAM fmstrs_nl.f

TABLE B-13. TYPICAL DEBUGGING FILE FOR PROGRAM efsize.f

FILENAME efsize.log

crack_run38.bdf

 7.50000 -2.62500 .00000

feam_mdl.fem

crack_run38.f06

.063

-1.

1

NASTRAN global
Model

Origin of the local
model

Filename of the local
model

Filename of the
NASTRAN solution

Thickness of the skin

Directional vectors of the
bar elements

TABLE B-14. TYPICAL OUTPUT FILE FOR PROGRAM fmsts_nl.f

FILENAME crack_run72.bds

```

crack_run72.f06
No. eelm = 46
Sizes = -1.406   1.406   -1.406   1.406
Cenetr = 13.50000 -2.62500 .00000 .00000
Point 1 = 12.09375 -4.03125 .00000 .00000
Point 2 = 14.90625 -4.03125 .00000 .00000
Point 3 = 14.90625 -1.21875 .00000 .00000
Point 4 = 12.09375 -1.21875 .00000 .00000

```

STRESS AT THE NODAL POINTS

	4			
-1.50000	-.18750	.000	-5.15457E+03	1.04874E+03
-1.31250	-.18750	.000	-5.21425E+03	8.67914E+02
-1.31250	-.00500	.000	-1.05827E+04	-9.03662E+02
.
.
.
-1.50000	-.00500	.000	-1.05230E+04	-7.22836E+02

	4			
36877	4			
-1.31250	1.52500	.000	-1.65174E+03	1.97516E+03
-1.50000	1.52500	.000	-1.78161E+03	1.58163E+03
.
.
-1.50000	-.00500	.000	-1.05230E+04	-7.22836E+02

DISPLACEMENTS AT THE NODAL POINTS

	4			
36527	4			
-1.50000	-.18750	.000	-1.18186E-02	2.66353E-02
-1.31250	-.18750	.000	-1.19187E-02	2.76344E-02
-1.31250	-.00500	.000	.00000E+00	.00000E+00
-1.50000	-.00500	.000	.00000E+00	.00000E+00
.
.
-1.50000	-.00500	.000	-1.05230E+04	-7.22836E+02

	4			
36527	4			
-1.50000	-.18750	.000	-1.18186E-02	2.66353E-02
-1.31250	-.18750	.000	-1.19187E-02	2.76344E-02
-1.31250	-.00500	.000	.00000E+00	.00000E+00
-1.50000	-.00500	.000	.00000E+00	.00000E+00
.
.
-1.50000	-.00500	.000	-1.05230E+04	-7.22836E+02

	4			
Bar ID	X	Y	Z	Vx
5129	.000	.000	.000	3.49524E+02
5253	-.750	.875	.000	-1.42359E+03
.
.
-1.50000	-.00500	.000	-1.05230E+04	-7.22836E+02

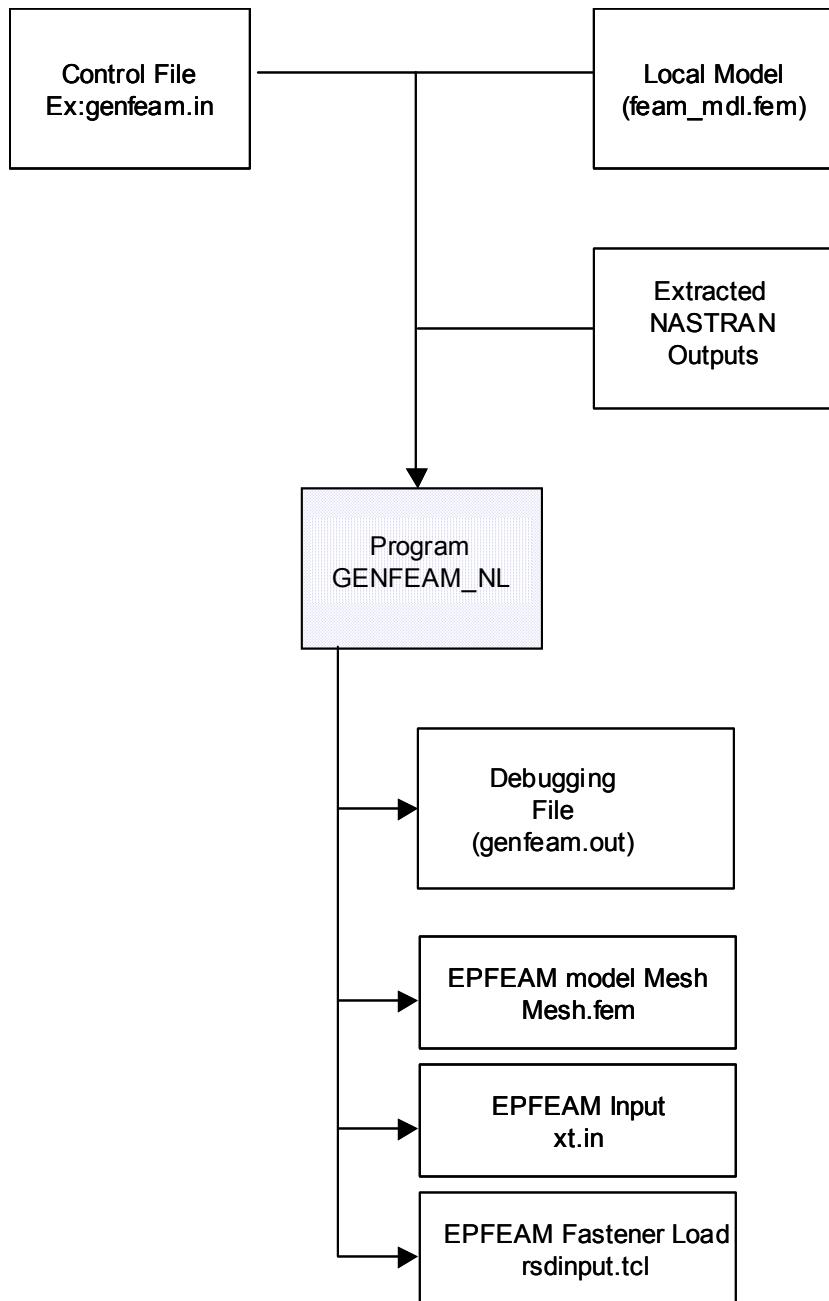


FIGURE B-7. INPUT AND OUTPUT FILES FOR PROGRAM genfeam_nl.f

TABLE B-15. TYPICAL INPUT FILE FOR PROGRAM genfeam_nl.f

FILENAME genfeam.in

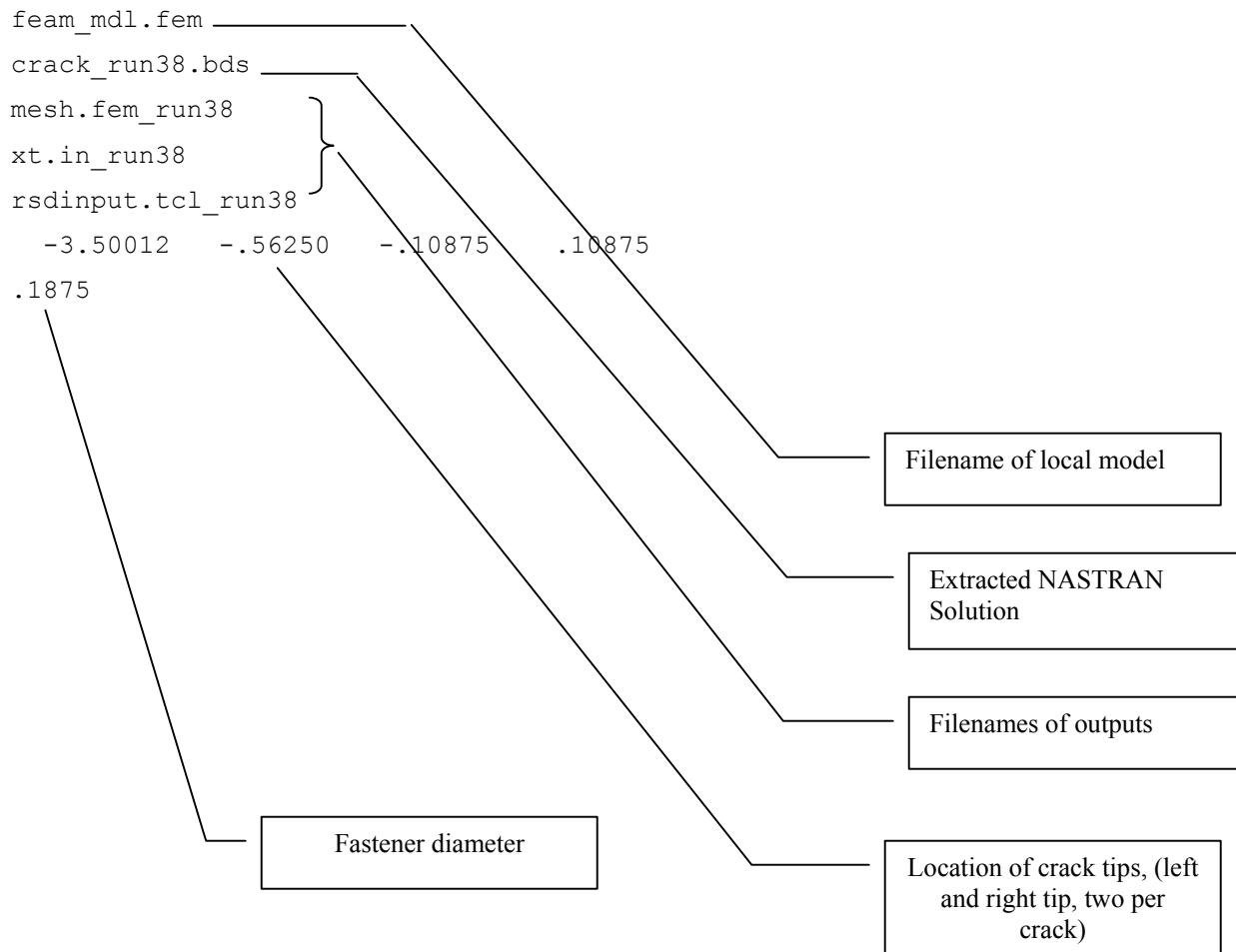


TABLE B-16. TYPICAL OUTPUT FILE FOR PROGRAM genfeam_nl.f

FILENAME xt.in

2
-5.50012 -1.31250 }
-.10875 .10875 }
-2.50012 1.00000 .00000
-2.50012 1.00000 .00000

Number of crack tips

Crack tip coordinates

Coordinates of the lower left corner

TABLE B-17. TYPICAL OUTPUT FILE FOR PROGRAM genfem_nl.f—DISPLACEMENTS AND TRACtions

FILENAME mesh.fem

Definition of tractions at the other three edges (T_x and T_y , three per element)

TABLE B-18. TYPICAL OUTPUT FILE FOR PROGRAM genfeam_nl.f

FILENAME mesh.fem

```
set crks {  
{ -5.50012 -1.12500 }  
{ -.10875 .10875 }  
}  
set mainCrk 2;  
set ALTCrkBodySize { -3.20016 5.76000 };  
set pressure 1;  
set KIC 120.0;  
set ALTPointLoad {  
{ -3.00000 .09375 -2.04721E-02 -7.72216E-03 }  
{ -1.50000 .09375 -2.86042E-02 1.89726E-03 }  
{ .00000 .09375 -2.84288E-02 5.54438E-02 }  
{ -3.00000 .87500 -9.35144E-02 -4.89712E-02 }  
{ 5.25000 .87500 2.40198E-01 5.21580E+00 }  
}
```

Coordinates of the crack tips

X coordinates of the left and right edges

Location and magnitudes of the concentrated loads (X, Y, Px, Py)

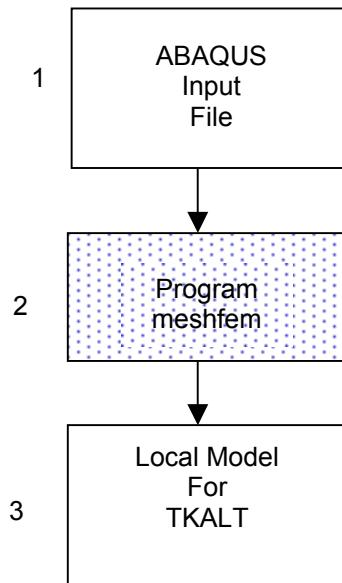


FIGURE B-8. INPUT AND OUTPUT FILES FOR PROGRAM meshfem.f

TABLE B-19. TYPICAL ABAQUS FILE FOR THE LOCAL MODEL

***HEADING**
ABAQUS job created on 23-Mar-98 at 15:21:57

*NODE
 1, 0.512501, 0.0249999
 2, 0.511724, 0.0374302
 3, 0.498601, 0.0373719
 4, 0.499376, 0.0249688
 5, 0.525001, 0.0499999
 6, 0.497519, 0.049752
 7, 0.525001, 0.0249999
 8, 0.51822, 0.101406
 9, 0.489483, 0.10201

Definition for the grid points

**
**
** fem
**

*ELSET, ELSET=FEM

1,	2,	3,	4,	5,	6,	7,	8,
9,	10,	11,	12,	13,	14,	15,	16,
17,	18,	19,	20,	21,	22,	23,	24,
25,	26,	27,	28,	29,	30,	31,	32,
33,	34,	35,	36,	37,	38,	39,	40,
41,	42,	43,	44,	45,	46,	47,	48,

Definition for the elements

TABLE B-20. TYPICAL TKALT FILE FOR THE LOCAL MODEL

1	-.98438	.02197					
2	-.97552	.02904					
3	-.98438	.02930					
4	-.98438	.03662					
5	-.09237	.01603					
6	-.09577	.01652					
7	-.10600	.01834					
8	-.97556	.01452					
9	-.98438	.01465					
10	-.98438	.00732					
11	-.97549	.04357					
12	-.98438	.04395					
13	-.96655	.05758					
14	-.98438	.05859					
15	-.98438	.05127					
16	-.98438	.07324					
17	-.96676	.00720					
18	-.96674	.01440					
<hr/>							
-1	.00000	.00000					
12	3	21	22	4	2	25	11
18	21	3	9	20	2	1	8
86	18	9	83	17	8	10	82
28	14	92	32	16	13	34	27
14	12	22	92	15	11	23	13
<hr/>							
<hr/>							
2863	2915	2866	2857	2865	2867	2859	2858
2914	2918	2915	2863	2920	2921	2865	2864
-1	0	0	0	0	0	0	0
-1	0	.00000	.00000				
0	-4	.000	.000	.000	.000	.000	.000
-1	0	.000	.000	.000	.000	.000	.000

Definition for the grid points

Termination for the nodes

Definition for the elements

Termination for the elements

Termination for the displacements and tractions

B.3 PROGRAM INSTRUCTION FOR tkalt.

This section lists the basic assumptions used in the tkalt code, the input parameters, examples of the tool command language (TCL) control files, and the executable TCL file for program flow control.

B.3.1 ASSUMPTIONS THAT ARE HARD CODED IN tkalt.

1. The lead crack tip is on the left-hand side of the local region, modeled as an edge crack.
2. There is only one lead crack.
3. runID increases as the coordinate of the lead crack tip increases
4. Boundary conditions (BC) are stored in the files \$bcFileRoot\$runID
5. rsdinput's are \$rsdinputRoot\$runID
6. Point loads are specified in \$rsdinputRoot\$runID
7. Input files are in the directory \$inputDir
8. BC are given at unit load level
9. xmin > -999 for the local model
10. Lead crack is extended by a fixed increment when the runID is increased by 1
11. The following parameters are hard code in the procedure addTstar in the executable module.

contour size: epsilon = 0.087"
crack growth step: crkIncr = 0.04"
load increment step: loadIncr = 3%

12. The strain-stress curve is hard coded in the procedure addFlowCurve in the executable module, if the TCL variable myPlasticity is set to yes.

B.3.2 GLOBAL TCL VARIABLES.

1. Specify the paths and names for the input files. The input files include the following:

- a. A finite element method (FEM) model file

This specifies the FEM mesh used by the EPFEAM. It contains the coordinate of nodes and nodes of elements.

- b. A number of boundary condition (BC) files

This specifies the prescribed nodal displacements and traction boundary conditions.

- c. TCL command file—This specifies the TCL variables, such as:

ALTCrkBodySize <the max. and min. x-coord. of the local model>
ALTPointLoad <point loads>, etc.

Examples for other parameters:

```
set bcFileRoot      mesh.fem_run
set rsdinputRoot    rsdinput.tcl_run
set femModel        feam_mdl.fem
set inputDir        .
set curRunID        75
set lastRunID       113
```

Here,

The input files are located in the current directory feam_mdl.fem is the name of the FEM model file. mesh.fem_run is the root name of BC files. So, the boundary conditions are stored in

```
mesh.fem_run75,
mesh.fem_run76,
...,
mesh.fem_run113.
```

Associated files for TCL commands are

```
rsdinput.tcl_run75,
rsdinput.tcl_run76,
rsdinput.tcl_run113.
```

- 2. The size of multiple-site damage (MSD) cracks.

Two equal length MSD cracks are assumed to be emanating from the edge of a rivet hole. All the MSD cracks are assumed to have the same initial crack length. This variable will be used, together with others, to construct the list of crack positions. The following command specifies that MSD cracks have an initial crack length of 0.05".

```
set MSDsize 0.05
```

3. T*-integral resistance curve

The TCL command that will give the T*-integral value on the resistance curve for a given amount of crack growth da.

Example 1:

```
set myTStar {1.1*<0.7063-0.03982*$da-0.06973/<0.1746+$da>>}
```

In the above example, the T*-integral resistance curve is evaluated using a TCL expression.

Example 2:

```
set myTStar {[tStarProc $da] }
proc tStarProc { da } {
    if { $da < 0.09 } {
        return 0.53
    } else {
        return [expr 1.35*<0.7063-0.03982*$da-0.06973/<0.1746+$da>>]
    }
}
```

In example 2, the T*-integral resistance curve is evaluated by calling a TCL procedure, tStarProc. When the amount of crack growth is smaller than 0.09", the T*-integral on the resistance curve is 0.53 <ksi in>; otherwise, it is given as $1.35 * \langle 0.7063 - 0.03982 * \$da - 0.06973 / \langle 0.1746 + \$da \rangle \rangle$.

4. Position of the lead crack tip

Example:

```
set curLeadTip      -0.563
set curRunLimit     -0.563
set runLimitIncr   0.09375
```

Here, it specifies that the initial position of the tip of the lead crack is at -0.563. The boundary condition for the first run case, given in the files \$inputDir/\$bcFileRoot\$curRunID, and \$inputDir/\$rsdinputRoot\$curRunID, are valid until the lead crack tip passed -0.563. The boundary condition in the next run case will be valid until the lead crack passes $-0.563 + 0.09375$.

5. The x coordinates of the rivet holes with MSD crack

Lead cracks must be on the x axis. Equal length MSD cracks can be specified on the rivet holes ahead of the tip of the lead crack.

Example:

```
set holePosition {0 1.5 3.0}
```

Here, three rivet holes, centered at x=0", x=1.5", x=3.0" have MSD cracks, with an initial crack length = \$MSDsize. Note, the x coordinates must be presented in the ascent order in the list.

6. The size of rivet holes

The radius of the rivet holes.

Example:

```
set holeRadius 0.082
```

Two MSD emanating from a rivet hole are modeled as a single small crack from

x= \$holePosition-<\$holeRadius+\$MSDsize>

to

x= \$holePosition+<\$holeRadius+\$MSDsize>.

7. Optional Inputs: no rivet force on the crack surface on

Example:

```
set noInitCrackSurfaceRivet yes
set pinY 0.082
```

In this case, the program will try to remove the point loads applied at y=0.082 and x< \$curLeadTip, specified in the input files rsdinput.tcl_run*. When noInitCrackSurfaceRivet and pinY are not specified, no point load will be removed.

8. Input/output files and directories

Example:

```
set caseID 1
set no 2
set case case$caseID
set homeDir [file join $env<HOME> caseEp/$case/res/$no]
source [file join $env<HOME> caseEp/$case/x.conf]
```

In the above example, the results <out.dat, tear.dat> will be saved in the directory \$homeDir, i.e.,

~/caseEp/case1/res/2/;

some of the TCL commands are in ~/caseEp/case1/x.conf. Note, it may be convenient to save some of the common TCL commands in one file, say,

~/caseEp/case1/x.conf,

so that it can be reused by different runs of tearing analysis.

Note: \$caseID will be used by \$getBCCmd

9. Using BC at specific load level—Nonlinear global Analysis.

When the lead crack tip reaches a crack length that requires BC update, it will invoke \$getBCCmd to get BC at current load level. \$getBCCmd is executed with three arguments:

\$getBCCmd caseID currentRunID currentLoadLevel

caseID is the identification number of the case

currentRunID is the run ID for which the BC is requested. currentLoadLevel is the predicted load level at current crack length. The boundary condition for the local region at this level is requested. However, the boundary condition must be linearly scaled to that for the unit level in the input files \$bcFileRoot\$runID and \$rsdInputRoot\$runID.

If getBCCmd is not specified, it is assumed that the BC is precalculated and stored in the directory for input files, i.e.,

<\$inputDir>

When getBCCmd is specified, it will be invoked with the three arguments to get BC at run time. \$getBCCmd must generate the BC and save the input files in \$inputDir.

Example:

```
set getBCCmd /uhs1112013/d1xr/c045270/Bin/getbndy
```

Note: The environment variable PATH must be set correctly, if \$getBCCmd invokes other unix commands.

10. Specify strain-stress curve

This command specifies the strain-stress curve in the files \$rsdinputRoot\$runID.

Example:

```
ALTResource matID CurveFitted
ALTResource flowCurve {
 0.0043 45
 0.01 51.5
 0.02 56
 0.04 61
 0.07 65.5
 0.10 68
 0.16 70
}
```

In this case, Young's modulus will be determined from the first pair of strain-stress data.

11. Advanced Features

The user may use the file xt.tcl at the current directory to dynamically control the execution of tkalt. The executable module will execute the commands in xt.tcl after each step of stable tearing analysis, if xt.tcl exists at the current directory. See the executable module for details.

TABLE B-21. TCL EXAPMLE FOR RUNNING MSD FLAT PANEL WITH
0.100" MSD CRACKS

Filename.run.tcl

```

define the location for the input files

set inputDir .
set bcFileRoot    mesh.fem_run
set rsdinputRoot  rsdinput.tcl_run
set femModel      feam_mdl.fem

# the T* curve is saved in tStar.tcl

source [file join $env<HOME> T_star_Rst/tStar.tcl]

# define the BC set.

set curRunID          44
set lastRunID         67
set curRunLimit       -0.7125
set runLimitIncr     0.1425
set curLeadTip        -0.40975

# define rivet hole and MSD

set holeRadius         0.09375
set holePosition       {0 1.14 2.28}
set pinY               0.09375
set MSDsize             0.100

# where to put the output file

#global caseID
set caseID 2
set no 3
set case case$caseID
set homeDir [file join $env<HOME> test]
set homeDir [file join $env<HOME> Trun/Run_203 ]

set nonlinearInit yes

# how to get BC

set getBCCmd ~/Bin/getbndy

#run executable module ( see Appendix XX for complete listing)

source $AltLibDir/tearApp.tcl

```

TABLE B-22. TCL EXAPMLE FOR T*-INTEGRAL RESISTANCE CURVE

File name : T_star_Rst/tStar.tcl

```
set myTStar { [tStarProc $da] }
proc tStarProc { da } {
    if { $da < 0.09 } {
        return 0.53
    } else {
        return [expr 1.35*<0.7063-0.03982*$da-0.06973/<0.1746+$da>>]
    }
}
```

TABLE B-23. FORMAT OF TCL COMMAND IN THE run.tcl FILE

```
begin run.tcl
    set inputDir      <variable>
    set curRunID     <variable>
    set lastRunID    <variable>
    bcFileRoot       <variable>
    rsdinputRoot    <variable>
    femModel        <variable>
    inputDir         <variable>
    curRunID         <variable>
    lastRunID        <variable>
    MSDsize          <variable>
    MyTStar          <variable>
    curLeadTip      <variable>
    curRunLimit     <variable>
    runLimitIncr   <variable>
    holePosition    <variable>
    holeRadius      <variable>
    noInitCrackSurfaceRivet <variable>
    pinY            <variable>
    homeDir          <variable>
    getBCCmd         <variable>
    source $AltLibDir/tearApp.tcl
end of run.tcl
```

TABLE B-24. EXECUTABLE TCL COMMAND FOR THE EPFEAM ANALYSYS

FILENAME tearapp.tcl

```

catch { unset runIDDef }
set i $curRunID
set x $curRunLimit
while { $i <= $lastRunID } {
    set runIDDef<$i> $x
    incr i
    set x [expr $x+$runLimitIncr]
}
catch { unset i; unset x }

check the proc's
addMSD
addTstar
for hard coded parameters

if stiffness matrix can be reused,
set reuse 1
otherwise
set reuse 0

some unix system commands

set catCmd /usr/bin/cat
set mvCmd /usr/bin/mv
set cpCmd /usr/bin/cp

procedures

append two more MSD cracks

proc addMSD { } {
    hard coded

    global curLeadTip holePosition holeRadius MSDsize
    global jumpDef
    set jumpDef {}

    set the left crack tip at -999 so that it is out side the
    local model

```

```

set out "\nset crks {\n{-999 $curLeadTip}\n"
if { $MSDsize >= 0 } {
    set radius [expr $holeRadius + $MSDsize]
    foreach center $holePosition {
        set leftTip [expr $center - $radius]
        set rightTip [expr $center + $radius]
        lappend out " $leftTip $rightTip "
        append out "\n"
    }
} else {
    set radius [expr <$holeRadius < 0.095> ? 0.095 : $holeRadius]
    foreach center $holePosition {
        set leftTip [expr $center - $radius - 0.1]
        set rightTip [expr $center + $radius]
        append jumpDef " $leftTip $rightTip "
    }
}
append out "}\n"
return $out
}

```

specify T resistance curve*

```

proc addTstar { } {
    hard coded

    set out {
        set daLimit          1.30
        crack::config -epsilon .087
        crack::config -crkIncr .040
        crack::config -loadIncr   3
        set pressure 1
        set plotInterval 1
    }

    global myTStar
    if { [info exists myTStar] } {
        append out "crack::config -tStar [list $myTStar]\n"
    }
    return $out
}

```

flow curve for plasticity. Young's modulus will be determined from the first pair of strain stress data.

```

proc addFlowCurve { } {
    hard coded

    set out {
        ALTResource matID CurveFitted
        ALTResource flowCurve {
            0.0043 45

```

```

        0.01      51.5
        0.02      56
        0.04      61
        0.07      65.5
        0.10      68
        0.16      70
    }
    ALTResource
}
return $out
}

create mesh.fem in the current directory

proc genMeshFem { runID } {
    global inputDir bcFileRoot femModel
    global catCmd

    set model [file join $inputDir $femModel]
    set bc      [file join $inputDir $bcFileRoot$runID]
    exec $catCmd $model $bc > mesh.fem
}

create mesh.pt in the current directory

proc genMeshPt { runID } {
    global inputDir rsdinputRoot noInitCrackSurfaceRivet
    global pinY

    set noRivet 0
    if { [info exists noInitCrackSurfaceRivet] } {
        if { $noInitCrackSurfaceRivet } {
            set noRivet 1
        }
    }

    set ALTPointLoad {}
    source [file join $inputDir $rsdinputRoot$runID]
    set fileID [open "mesh.pt" w ]
    puts $fileID "set ALTPointLoad {}"
    if { $noRivet } {
        foreach pinLoad $ALTPointLoad {
            foreach {x y px py} $pinLoad {}}
            if { ! <$x < 0 && $y == $pinY> } {
                puts $fileID "{$pinLoad}"
            }
        }
    } else {
        foreach pinLoad $ALTPointLoad {
            puts $fileID "{$pinLoad}"
        }
    }
    puts $fileID "}"
    close $fileID
}

```

```
}
```

create rsdinput.tcl in the current directory

```
proc genRsdinputTcl { runID } {
    global inputDir bcFileRoot rsdinputRoot femModel myPlasticity
    global catCmd

    append content "source [file join $inputDir $rsdinputRoot$runID]"
    append content "\n"
    append content [addMSD]
    append content [addTstar]
    if { [info exists myPlasticity] } {
        if { $myPlasticity } {
            append content [addFlowCurve]
        }
    }
    append content "source mesh.pt"
    exec $catCmd > rsdinput.tcl << $content
}
```

convert mesh.fem into a binary file: xt.config

```
proc genBinInput { } {
    global AltBinDir
    global mvCmd
    exec $AltBinDir/genConf > local.mesh
    exec $mvCmd mesh.config xt.config
}
```

*check if the lead crack is reaching the edge of rivet hole
Assume, there is only one lead crack at left hand side of the panel*

```
proc crack:::checkJump {} {
    global jumpDef
    set crks $crack<list>

    set id [lindex $crks 0]
    set b $crack<$id,b>
    foreach {l r} $jumpDef {
        if { <$b > $l && <$b < $r > } {
            set crack<$id,b> $r
            set crack<$id,b0> $r
            break
        }
    }
}
```

customized routine to generate contours for grow cracks

```
proc crack:::genContour { file } {
    global stopMSD
    set noMSD 0
```

```

if { [info exists stopMSD] } {
    if { $stopMSD } {
        set noMSD 1
    }
}

set crks $crack<list>

set leftEdge 0
set rightEdge 0
switch $crack<edgeCrk> {
    left { set leftEdge 1 }
    right { set rightEdge 1 }
}

set num [expr [llength $crks]*2]
if { $leftEdge || $rightEdge } {
    incr num -1
}

set fileID [open $file w]
puts $fileID $num
set i 0
set num [expr [llength $crks]-1]

pick a good guess for numpt

set rad $crack<epsilon>
set step $crack<contourStep>
set halfnum [expr int<$rad/$step>]
set halfnum2 [expr 2*$halfnum]
while { $i <= $num } {
    set id [lindex $crks $i]

    if { $i == 0 } {
        set left [ expr $crack<$id,b0>-$rad ]
        set right [ expr $crack<$id,b>+$rad ]
        set numpt [ expr int<<$right-$left>/\$step>]
        global curLeadTip
        set curLeadTip [ expr $crack<$id,b>+0.0 ]
        puts $fileID "           $left      -0.0001"
        puts $fileID "$halfnum   $left      -$rad"
        puts $fileID "$numpt     $right     -$rad"
        puts $fileID "$halfnum2  $right     $rad "
        puts $fileID "$numpt     $left      $rad "
        puts $fileID "$halfnum   $left      0.0001"
        puts $fileID "0 0 0\n"
    }

} else {
    if { $noMSD } {
        set left [ expr $crack<$id,a>-$rad ]
        set right [ expr $crack<$id,a0>+$rad ]
        set numpt [ expr int<<$right-$left>/\$step>]
        puts $fileID "           $right     -0.4"
        puts $fileID "0 0 0\n"
    }
}

```

```

set left [ expr $crack<$id,b0>-$rad ]
set right [ expr $crack<$id,b>+$rad ]
set numpt [ expr int<<$right-$left>/\$step>]
puts $fileID "           \$left      -0.4"
puts $fileID "0 0 0\n"

} else {
    set left [ expr $crack<$id,a>-$rad ]
    set right [ expr $crack<$id,a0>+$rad ]
    set numpt [ expr int<<$right-$left>/\$step>]
    puts $fileID "           \$right -0.0001"
    puts $fileID "$halfnum \$right -$rad "
    puts $fileID "$numpt \$left -$rad "
    puts $fileID "$halfnum2 \$left \$rad "
    puts $fileID "$numpt \$right \$rad "
    puts $fileID "$halfnum \$right 0.0001"
    puts $fileID "0 0 0\n"

    set left [ expr $crack<$id,b0>-$rad ]
    set right [ expr $crack<$id,b>+$rad ]
    set numpt [ expr int<<$right-$left>/\$step>]
    puts $fileID "           \$left      -0.0001"
    puts $fileID "$halfnum \$left      -$rad "
    puts $fileID "$numpt \$right     -$rad "
    puts $fileID "$halfnum2 \$right \$rad "
    puts $fileID "$numpt \$left      \$rad "
    puts $fileID "$halfnum \$left      0.0001"
    puts $fileID "0 0 0\n"
}
}

incr i
}
close $fileID

checkBC
}

proc checkBC { } {
    global curRunID runIDDef curLeadTip

    Assume that runID increases as the coordinate of the
    lead crack tip increases

    global BCChanged
    if { ! [info exists BCChanged] } {
        set BCChanged 0
    }

    while { 1 } {
        if { ! [info exists runIDDef<$curRunID>] } {

            runID not defined. No boundary condition
            exists for this case. Quit.

            puts " *** stop tearing at runID = $curRunID ***"

```

```

        set BCChanged 0
        quitTearing
        break
    } else {
        set limit $runIDDef<$curRunID>
        if { $curLeadTip > $limit } {
            incr curRunID
            set BCChanged 1
        } else {
            break
        }
    }
}

if { $BCChanged } {
    InterBC
}
}

proc InterBC {} {
    global getBCCmd

    if { [info exists getBCCmd] } {
        set cmd /uhs1112013/d1xr/c045270/Bin/getbndy
        set cmd $getBCCmd

        global curLoadLevel curRunID caseID
        exec $cmd $caseID $curRunID $curLoadLevel
    }

    genMeshFem $curRunID
    genMeshPt $curRunID
}

proc getInitBC {} {
    checkBC
    global BCChanged
    if { ! $BCChanged } {
        InterBC
    } else {
        set BCChanged 0
    }
    catch {
        global daLimit
        if { $daLimit < 0 } {
            puts " *** stop: not init BC found"
            exit
        }
    }
}

```

*convert out.dat into tear.dat. tear.dat contains load level versus x-coordinates of the crack tips.
The format of tear.dat is*

```

loadLevel1 crackTip1 crackTip2 crackTip3 ...
loadLevel2 crackTip1 crackTip2 crackTip3 ...
...
.

proc listLoadLevel { } {
    set fileID [open "out.dat" r]
    set tearID [open "tear.dat" w]
    set n 0
    while { ! [eof $fileID] } {
        gets $fileID line
        if { ! [regexp -- "load level: <<[0-9.\"]*>>" $line match load] } {
            continue;
        }

        read off the comment for offset

        gets $fileID line
        set tips $load
        set i 0
        while { 1 } {
            gets $fileID line
            if { [regexp -- "" $line] } {
                while { $i < $n } {
                    incr i
                    lappend tips $ptip
                }
                set n $i
                break;
            }
            set tip [lindex $line 0]
            if { $tip != {} } {
                incr i
                set ptip $tip
                lappend tips $tip
            }
        }
        puts $tearID $tips
    }
    close $tearID
    close $fileID
}

```

1. Result will be added to out.dat incrementally
2. If daLimit is specify, the termination of the tearing simulation will be controled by daLimit only. Otherwise, it terminates when all cracks are linked up.
3. Add a run time control - a tcl script file: xt.tcl
It will be sourced, if exists at the run time,
 1> after residual strength analysis for the initial cracks
status = AfterRsd
 2> after each tearing step <increase the load or grow the crack>
status = InTearing

```

proc ALTear:::App { {plotItems {}} {anaReuse 0} {anaGenLoc 0} } {
    crack:::GetTstar ALTCrkTstar ALTCrkEpsilon .getTstar
    global AltLibDir AltBinDir
    global plotInterval
    if { ! [info exists plotInterval] } {
        set plotInterval 0
    }

    global daLimit
    if { ! [info exists daLimit] } {
        set daLimit 1000000
    }
    set curGrowStep 0
    set plotID 0

    catch {global crack; unset crack}
    if { $anaGenLoc == 0 } {
        ALTRsd:::makeMeshCtrl
        ALTRsd:::preProc
    }
}

must specify the body size first

global ALTPointLoad; set ALTPointLoad {}
source rsdinput.tcl
ALTRsd:::GenPtLoadFile
crack:::config -bodySize "$ALTCrkBodySize"

foreach crk $crks {
    eval new crack $crk
}

set da 0
global crack
set Jo [expr $crack<tStar>]
set KIC [expr sqrt<10500*$Jo>]

puts "KICo= $KIC"
puts "tStar=$crack<tStar>"
puts "epsilon=$crack<epsilon>"
puts "crkIncr=$crack<crkIncr>"
puts "loadIncr=$crack<loadIncr>%"

crack:::genTipFile xt.in
crack:::genContour xt.J1
set rsd [ new ALTRsd $KIC $pressure]
set eload [ALTRsd:::elastic $rsd $anaReuse]
set pload [ALTRsd:::plastic $rsd]

if { [file exists "xt.tcl"] } {
    set status {AfterRsd}
    source "xt.tcl"
}

```

```

set iniLdFactor [expr $pload/$pressure]
ALTRsd::preProc $iniLdFactor

note: replaced with scaling after ALT::Init

set ALTOrigLoad    $pressure
set ALTLoad        $pload
set ALTLoadFactor 1
set ALTLoadIncr   [expr 0.01*$crack<loadIncr>]
set ALTLoadPlotOffSet 0

crack::genTipFile xt.in
crack::genContour xt.J1

set fileID [open "out.dat" w]
puts $fileID "\ the history of crack growth"
puts $fileID "\$data=curve2D"
close $fileID

==== analyze ====
global curLoadLevel
set curLoadLevel [expr $iniLdFactor*$ALTLoadFactor*$ALTOrigLoad]
InterBC

ALT::Init
exec $AltBinDir/genBC [expr $iniLdFactor] xt.bnd
ALTRsd::GenPtLoadFile [expr $iniLdFactor]
ALT zeroEdgeLoads readNewBC saveBC; new BC

delete [ ALT::SetUp -noAssembly]
delete [ ALT::Elastic -10 ]
set vT [ ALT::FullPlastic 15 ]
ALT::SaveState
eval ALT::Plot $plotItems

set analyzing 1
while { $analyzing } {
    puts "Load Level: [expr $iniLdFactor*$ALTLoadFactor*$ALTOrigLoad] \n"
    crack::printAll

    set vT [crack::fixValue $vT]
    puts "T*-integral:"
    set fileID [open "out.dat" a]
    ALTTear::printCrack $fileID \
        [expr $iniLdFactor*$ALTLoadFactor*$ALTOrigLoad] $ALTLoadPlotOffSet
    vec::print $vT
    close $fileID

    if { [file exists "xt.tcl"] } {
        set status {InTearing}
        source "xt.tcl"
    }

    if { [crack::grow [vec::value $vT]] } {

```

```

a hack to jump the crack from left edge to right edge
of a rivet hole

crack:::checkJump
if { $curGrowStep < $plotInterval } {
    incr plotID
    exec cp xt.stat stat.$plotID
}
incr curGrowStep
if { $curGrowStep >= $plotInterval } {
    set curGrowStep 0
}

set ALTLoadPlotOffset [expr 0.0015+$ALTLloadPlotOffset]
crack:::linkUp
puts " *** MSG ***: crack grow"
crack:::genTipFile xt.in
crack:::genContour xt.J1
ALT:::EpCrackGrowInit

a hack to allow change of BC in crack:::genContour
new BC will be in mesh.fem, mesh.pt and must set

    global BCChanged; set BCChanged 1

        to trig the update as the following.
global BCChanged
if { ! [info exists BCChanged] } {
    set BCChanged 0
} elseif { $BCChanged == 1 } {
    set BCChanged 0
    exec $AltBinDir/genConf > local.mesh
    exec mv mesh.config xt.config
    source mesh.pt
    exec $AltBinDir/genBC [expr $ALTLloadFactor*$iniLdFactor] xt.inc
    ALTRsd:::GenPtLoadFile [expr $ALTLloadFactor*$iniLdFactor]
    ALT:::LoadGrowInit xt.inc
}
} else {
    set ALTLloadPlotOffset 0
    puts " *** MSG ***: more load"
    set ALTLloadFactor [expr $ALTLloadFactor+$ALTLloadIncr]

    set curLoadLevel [expr $iniLdFactor*$ALTLloadFactor*$ALTOrigLoad]
    InterBC

    exec $AltBinDir/genBC [expr $ALTLloadFactor*$iniLdFactor] xt.inc
    ALTRsd:::GenPtLoadFile [expr $ALTLloadFactor*$iniLdFactor]
    ALT:::LoadGrowInit xt.inc
}

set vT [ ALT:::FullPlastic 15 ]
ALT:::SaveState
eval ALT:::Plot $plotItems

if { <[crack:::num] < 2> && <$daLimit >= 100000> } {
    puts " *** stop tearing at all cracks are linked up ***"

```

```

        set analyzing 0
    }
    if { [crack:::maxDa] > $daLimit } {
        if { $daLimit > 0 } {
            puts " *** stop tearing at da = [crack:::maxDa] ***"
        }
        set analyzing 0
    }
}
set fileID [open "out.dat" a]
ALTear:::printCrack $fileID [expr
$iniLdFactor*$ALTLoadFactor*$ALTOrigLoad]
close $fileID
}

```

support for run time control

1. *loadTk*: load the top window . and
2. *showPlotMenu*: show the menu for plotting
Tearing simulation will be paused until exiting the menu
3. *quitTk*: quit tk and resume tearing simulation
4. *quitTearing*: quit tk and ready to stop tearing simulation

An Example of *xt.tcl*. It brings up the menu for plotting. The tearing analysis is paused. It will resume after the plotting session.

```

loadTk
showPlotMenu {Example}

```

Another example. It bring the tearing analysis to an end.

```

break

```

One more. The user can put in more commands in file *yt.tcl* to be executed after the plotting. *yt.tcl* can be written just before the user quit the plotting menu.

```

loadTk
showPlotMenu {Wait and see}
if { [file exists "yt.tcl"] } {
    source "yt.tcl"
}

proc loadTk { } {
    global AltLibDir
    catch {
        StartTk $AltLibDir/ALTrc.tk
        destroy .
        destroy .
    }
    if { [info command exitAll] == {} } {
        rename exit exitAll
    }
}

```

```

proc exit { } {
    rename exit {}
    rename exitAll exit
    global runAfterTk
    set runAfterTk 1
    if { [info command .t] != {} } {
        destroy .t
    }
}
if { [info command .l] == {} } {
    pack [label .l -width 0 -height 0 -text "tearing ..."]
}
}

proc showPlotMenu { {status {tearing ...}} } {
    if { [info command .l] != {} } {
        pack forget .l
    }
    catch {
        destroy .h
        destroy .f
    }
    global anaType anaTypeID
    set anaTypeID 6
    set anaType "Stable Tearing"
    pack [frame .t]
    pack [label .t.h -width 50] -fill x -expand true \
        -ipady 1 -ipadx 3 -side bottom
    getVisualOpt {.t} .t.h

    waitTk
    destroy .t
    tkwait window .t

    if { [info command .l] == {} } {
        pack [label .l -width 0 -height 0 -text $status]
    } else {
        .l configure -text $status
        pack .l
    }
    tkwait visibility .l
}

proc waitTk { } {
    global runAfterTk
    vwait runAfterTk
}

proc quitTk { } {
    if { [info command exitAll] != {} } {
        exit
    }
}

```

```

proc quitTearing { } {
    quitTk
    global daLimit
    set daLimit -1
}

show progress in runID

proc showServerScript { } {
    global port scaleCmd
    set port 2021
    set scaleCmd {.f.s}
    set out {
        proc showLine { channel } {
            gets $channel line
            if { [eof $channel] } {
                catch { close $channel }
                exit
            }
        }
        atch {eval $line}

        while { ! [fblocked $channel] } {
            gets $channel line
            if { [eof $channel] } {
                catch { close $channel }
                exit
            }
            catch {eval $line}
        }
        global sname
        set from [$sname cget -from]
        set to [$sname cget -to]
        set intv [expr "<$to-$from>/5"]
        $sname configure -tickinterval $intv
    }
    proc connect { channel addr port } {
        fconfigure $channel -blocking no
        fileevent $channel readable "showLine $channel"

        global serverID
        close $serverID
    }
}
append out "\"
if { \[ catch {\
    set serverID \[socket -server connect $port\] \} \] } {\ \
    exit \
} \
"
append out {
    pack [label .l -text "Stable Tearing"] -side top
    set f [frame .f -bd 2 -relief groove]
    global sname
}

```

```

}

append out "set $name $scaleCmd"
append out {
    scale $name -showvalue yes -width 5 -length 300 -variable status \
        -orient horizontal -from 0 -to 100 -resolution 1 -tickinterval 10 \
        -state disabled
    pack $name -side left
    pack $f -padx 7 -pady 2
    update
}
return $out
}

proc startStatusBar { } {
    global socketID
    if { ! [info exists socketID] } {
        set socketID {}
    }
    if { $socketID == {} } {
        exec wish << [showServerScript] &
    } else {
        return
    }
    after 100 { set socketID {} }
    vwait socketID

    global port
    set n 15
    while { $n > 0 } {
        if { [catch { set socketID [socket localhost $port] } ] } {
            after 100
            incr n -1
        } else {
            break
        }
    }
    if { $socketID == {} } {
        puts "fail to connect localhost $port"
        exit
    }
    initStatus
}

proc initStatus { } {
    global curRunID lastRunID socketID scaleCmd
    puts $socketID "$scaleCmd configure -from $curRunID -to $lastRunID"
    flush $socketID
    showStatus
}

proc showStatus { } {
    global curRunID scaleCmd socketID
    if { [catch {
        puts $socketID "global status; set status $curRunID";
        flush $socketID } ] } {
        catch {close $socketID}
        set socketID {}
    }
}

```

```

        }
    }

proc moveResults { } {
    global homeDir
    if { [info exists homeDir] } {
        file rename -force out.dat tear.dat $homeDir
    }
}

```

run a problem

```

global curLoadLevel
set curLoadLevel 1
    set pressure in addTstar to the same

```

```

getInitBC
genBinInput
genRsdinputTcl $curRunID

```

elastic case

ALTALT::Run 1 1

```

if { [info exists xt.tcl] } {
    source "xt.tcl"
}

```

tearing

```

puts "starting from runID: $curRunID"
puts "last runID: $lastRunID"

```

show status

```

if { [info exists env<DISPLAY>] } {
    if { $env<DISPLAY> != {} } {
        startStatusBar
    }
}

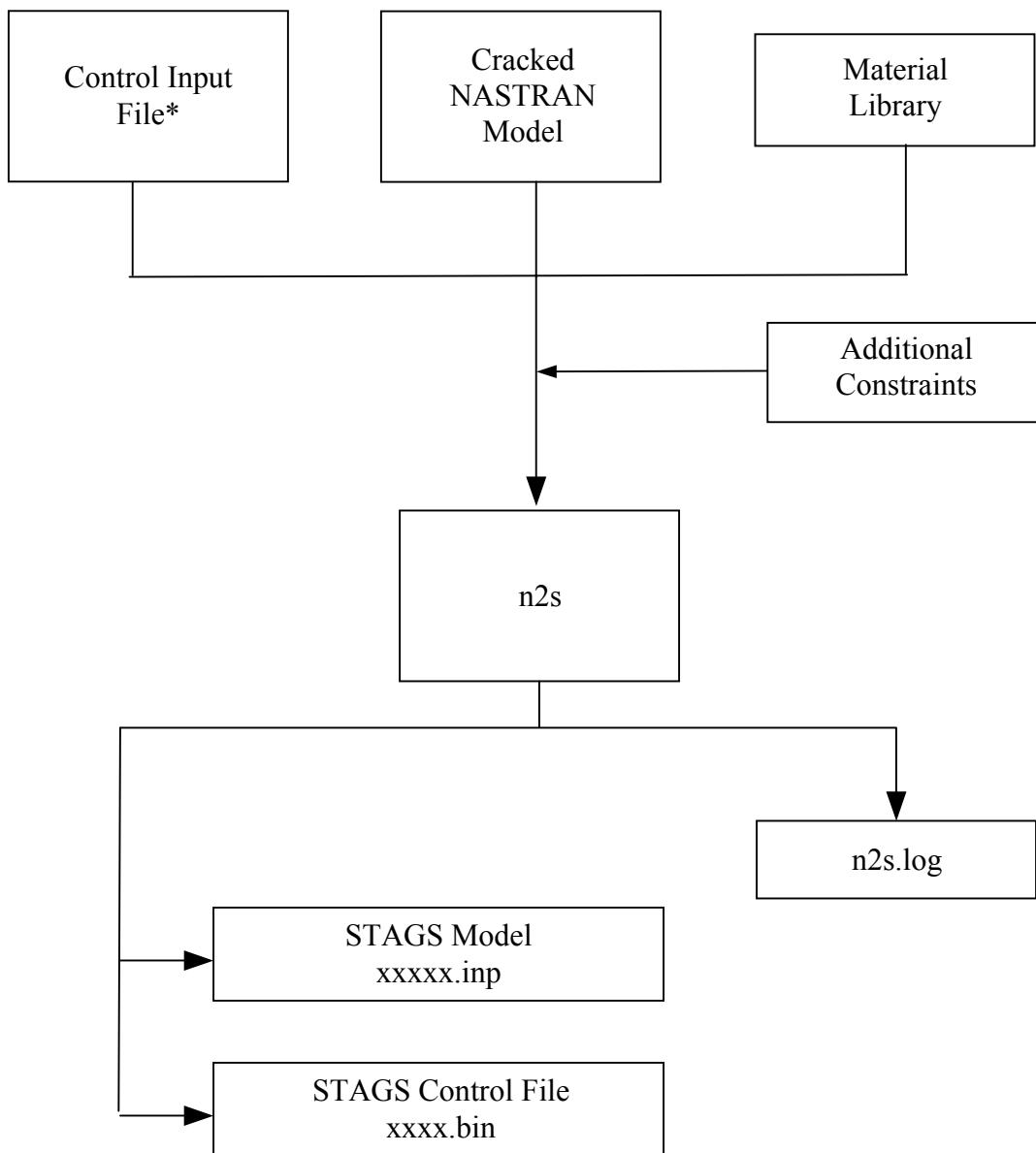
```

```

ALT Tear::App {} $reuse 1
listLoadLevel
moveResults
exit

```

B.4 COMPUTER CODE FOR NASTRAN-TO-STAGS CONVERSION.



*Note: See next page for input instruction

FIGURE B-9. FLOWCHART OF CONVERSION CODE n2s

TABLE B-25. INPUT FORMAT FOR n2s CODE

Line 1	\$inda
Line 2	j_title='.....'
Line 3	mat_fn='.....'
Line 4	nas_fn='.....'
Line 5	stags_fn='.....'
Line 6	crack_fn='.....'
Line 7	quad5_fn='.....'
Line 8	baref='.....'
Line 9	spcref='.....'
Line 10	elasps='....'
Line 11	mat_lib='....'
Line 12	g2card='....'
Line 13	linear=I
Line 14	load_set=I1, I2
Line 15	grdset='111000'
Line 16	idisp=I
Line 17	ilin=I
Line 18	ipshl5=I1,I2,.....
Line 19	ctoa=R1,R2
last line	\$end
j_title=	'Job Title max=80 characters' (optional)
mat_fn=	'material ID cross reference table, including definition of PSHELL for plane strain core' (optional)
nas_fn=	'filename of the NASTRAN bulk data'
stag_fn=	'filename of the output STAGS file'
crack_fn=	'filename of the file that defines the cracked nodes' (optional)
quad5_fn=	'filename of the file that defines the 5-node quad elements' (optional)
baref=	'filename of the file that defines the reference node of the bar elements' (optional)
elasps=	"filename that contains the property ID of the linear elastic shells (optional)
mat_lib=	'filename of the material property library' (optional)
g2_card=	'filename of the file defines rigid links'
linear	=1 for linear-elastic analysis =2 for nonlinear analysis

load_set= Load case ID for case A and case B, respectively

grdset= character string of 6 characters, consists of either '0' or '1', 0 = fixed and 1=freed, used for additional constraints applied to ALL grid points in DOF =1 through 6 (optional)

idisp= either '1' or '0', '1' indicates the applied forces will be treated as applied forced displacement and the reaction forces of grids with forced displacements will be printed out in .out2 file (optional)

ilin= if ilin=0, linear-elastic shell properties will be used otherwise elastic-plastic shell properties will be used (optional)

ipshel5 Up to 10 PSHELL ID, specifies which PSHELL that will be checked if the 5-node quads exist

CTOA= Up to 2 critical CTOA angle, for fatigue crack and saw cut simulation, respectively (optional)

TABLE B-26. EXAMPLE OF INPUT FILE FOR n2s

Filename n2s.in

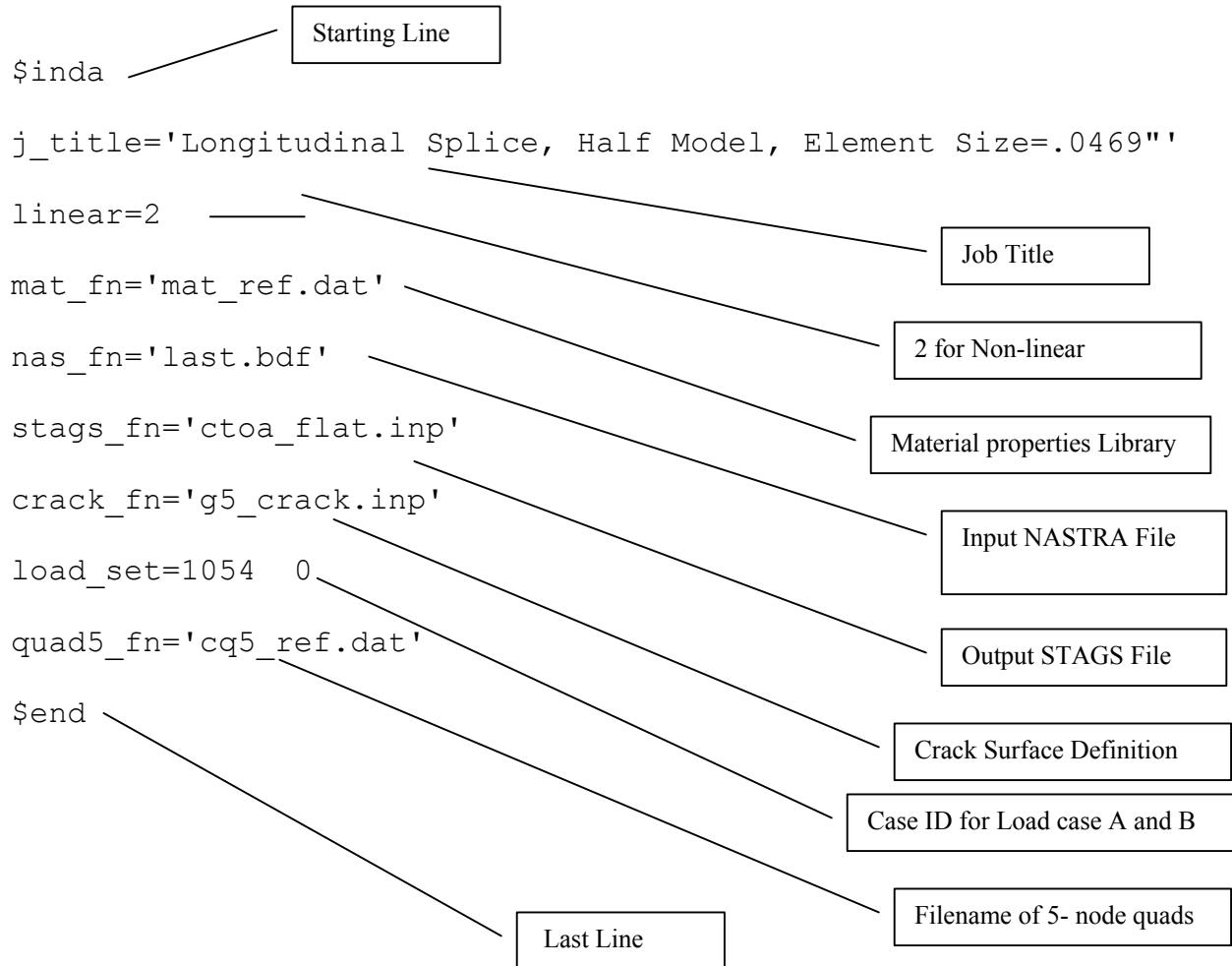


TABLE B-27. EXAMPLE OF INPUT FILE FOR n2s

Filename mat_ref.dat

2024 2002 0

20241 2002 1

7075 7075 0

1 - plane strain core
0 - plane stress

ID in the material Library

ID in the NASTRAN bulk data

TABLE B-28. INPUT FILE FOR n2s—MATERIAL PROPERTY LIBRARY

Filename for Material Library: mat_all.dat

```
=====
Clad 2024-T3 LT Sheet (Coupon test)
3001 1.05e+07 .33 3.94e6 0. 0.
9
.00282 29614
.00420 39327
.00512 43592
.00580 45487
.00660 46671
.01144 49159
.05600 58043
.11120 62781
.50000 63966
=====
```

Description of material

G (psi)

Poisson's ratio

Young's Modules (psi)

Material ID

Number of Strain-stress data point

Column 2 - Stress (psi)

Column 1 – Strain

```
=====
Clad 2024-T3 TL Sheet (Coupon test)
3002 1.05e+07 .33 3.94e6 0. 0.
9
.00265 27875
.00400 37017
.00480 40139
.00580 42146
.00920 46049
.01720 49728
.03920 54634
.10000 61324
.20200 62997
=====
```

```
=====
Clad 2024-T3 LT Sheet
2001 1.03e+07 .33 3.94e6 0. 0.
10
.0044 45320
.0050 47000
.0060 48500
.010 50500
.015 51700
.030 54800
.05 58000
.08 61200
.12 63200
=====
```

TABLE B-28. INPUT FILE FOR n2s—MATERIAL PROPERTY LIBRARY
 (Continued)

```

.184    63800
=====
Clad 2024-T3 TL Sheet, MIL-5 Fig. 3.2.1.1.6(b)
2002 1.03e+07 .33 3.94e6 0. 0.
10
.0036 37080.
.0046 41000.
.006  43800.
.010  46500.
.015  48600.
.03   52600.
.05   56400.
.08   59500.
.12   61800.
.18   63000.
=====
Clad 2024-T3 LT Sheet NASA data
2003 1.03e+07 .33 3.94e6 0. 0.
7
4.368900146e-03 4.500000000e+04
9.99999776e-03 5.150000000e+04
1.99999955e-02 5.600000000e+04
3.99999911e-02 6.100000000e+04
7.000000030e-02 6.550000000e+04
1.000000015e-01 6.800000000e+04
1.599999964e-01 7.000000000e+04
=====
Clad 2024-T3 TL Sheet, MIL-5 Fig. 3.2.1.1.6(b), Mod for large
disp
2004 1.03e+07 .33 3.94e6 0. 0.
10
.0036 37080.
.0046 41000.
.006  43800.
.010  46500.
.015  48600.
.03   52600.
.05   56400.
.08   59500.
.12   61800.
.50   65000.

```

TABLE B-28. INPUT FILE FOR n2s—MATERIAL PROPERTY LIBRARY
(Continued)

```
=====
7075-T6 LT Sheet - MIL-5 pg. 3-371
7075 1.03e+07 .33 3.94e6 0. 0.
8
.006505 67000.
.008 69000.
.0105 70000.
.026 72000
.060 75000.
.080 75800
.100 76000.
.200 76100.
=====
7075-T6 LT Sheet - MIL-5 pg. 3-371 , mod for large disp
7076 1.03e+07 .33 3.94e6 0. 0.
8
.006505 67000.
.008 69000.
.0105 70000.
.026 72000
.060 75000.
.080 75800
.100 76200.
.500 77000.
=====
Clad 2024-T3 LT Sheet , mod. for large disp
2005 1.03e+07 .33 3.94e6 0. 0.
10
.0044 45320
.0050 47000
.0060 48500
.010 50500
.015 51700
.030 54800
.05 58000
.08 61200
.12 63200
.500 64800
```

TABLE B-28. INPUT FILE FOR n2s—MATERIAL PROPERTY LIBRARY
 (Continued)

```
=====
Clad 2014-T6 LT Sheet MIL-5 Fig. 3.2.1.1.6(b)
2014 1.02e+07 .33 3.94e6 0. 0.
8
.005 51000.
.006 58200.
.010 62800.
.020 67000.
.040 70000.
.060 71200.
.100 71300.
.200 71400.
=====
Clad 2014-T6 LT Sheet MIL-5 Fig. 3.2.1.1.6(a) Longitudinal
2015 1.02e+07 .33 3.8356 0. 0.
9
0.00480392 49000.
.0053 52000.
.0060 54000.
.0070 56500.
.0100 58000.
.03 61000
.05 62000
.1000 63500
.2000 64000
=====
Clad 2014-T6 LT Sheet MIL-5 Fig. 3.2.1.1.6(a) Longitudinal mod.
upper strain
2016 1.02e+07 .33 3.94e6 0. 0.
8
.005 51000.
.006 58200.
.010 62800.
.020 67000.
.040 70000.
.060 71500.
.100 73000.
.200 74500.
```

TABLE B-28. INPUT FILE FOR n2s—MATERIAL PROPERTY LIBRARY
(Continued)

```
=====
Clad 2014-T6 LT Sheet MIL-5 Fig. 3.2.1.1.6(a) Longitudinal
2017 1.02e+07 .33 3.8356 0. 0.
9
0.00480392 49000.
.0053 52000.
.0060 54000.
.0070 56500.
.0100 58000.
.03 61000
.05 62000
.1000 64000
.2000 65500
=====
Clad 2014-T6 LT Sheet MIL-5 Fig. 3.2.1.1.6(a) Longitudinal mod.
upper strain
2018 1.02e+07 .33 3.94e6 0. 0.
9
.005 51000.
.006 58200.
.010 62800.
.020 67000.
.040 70000.
.060 71500.
.100 73000.
.200 74500.
.500 77000.
=====
Titanium
4001 1.60e+07 .33 3.94e6 0. 0.
3
.005 80000.
.010 160000.
.100 161000.
```

TABLE B-29. TYPICAL INPUT FILE FOR n2s

Format for Cracked Nodes: g5_input.dat

G5 cards (see STAGS input format)												
1	325	0	1	5.000	.040	.000	12.000	.000	100.000	\$	G-5	
0	1	12431	12431	1	0	1	15041	15041	1	0	\$	G-6
0	1	12430	12430	1	0	1	15040	15040	1	0	\$	G-
0	1	12429	12429	1	0	1	15039	15039	1	0	\$	G-6
0	1	12425	12425	1	0	1	15038	15038	1	0	\$	G-6
0	1	12333	12333	1	0	1	15037	15037	1	0	\$	G-6
0	1	12224	12224	1	0	1	15036	15036	1	0	\$	G-6
0	1	12082	12082	1	0	1	15035	15035	1	0	\$	G-6
0	1	12036	12036	1	0	1	15034	15034	1	0	\$	G-6
0	1	12035	12035	1	0	1	15033	15033	1	0	\$	G-6
0	1	12034	12034	1	0	1	15032	15032	1	0	\$	G-6
0	1	12033	12033	1	0	1	15031	15031	1	0	\$	G-6
0	1	11844	11844	1	0	1	15030	15030	1	0	\$	G-6
0	1	11843	11843	1	0	1	15029	15029	1	0	\$	G-g

G6 cards (see STAGS input format)											
-----------------------------------	--	--	--	--	--	--	--	--	--	--	--

TABLE B-30. EXAMPLE OF INPUT FILE FOR n2s—5-NODE QUAD REFERENCE DATA

Format for definition of the 5th node in the 5-node quad element (type 510): c5_ref.dat

3077	1571	1848	1908	1570	1484	2
3081	1569	1842	1907	1568	1482	2
3082	1560	1844	1904	1561	1476	2
3095	1566	1846	1906	1565	1480	2
3097	1564	1841	1905	1563	1478	2
3116	1592	1929	1878	1593	1927	1
3387	NASTRAN Element ID		609	618	373	1
4151	2		3398	3401	2607	1
50787	5815	5814	6209	6210	14616	3
45904	8932	8931	9294	9295	13265	1
45991	6897	6896	7314	7315	13267	3
50546	6638	6243	6246	6641	13698	1
50549	5855	5854	5857	6240	13668	4
50551	5851	5484	5854	5855	13633	4
50588	6639	6638	6641	7036	13696	1

Node ID (node 1-4)

Node ID of the 5th node

Edge number where the 5th node locates

The following is a description of the 2DFEAM code

All the commands used in the EPFEAM code are TCL commands. Knowledge about TCL is needed to take fully the advantage of the scripting commands provided.

CMC has a book on Tk/Tcl. There are also some tutorial material available on the internet. Dr. Kawai knows it.

1) ALTALT::Run -- run elastic analysis

ALTALT::Run anaReuse anaGenLoc

where anaReuse = 0 stiffness matrix can not be reused
= 1 can be reused
anaGenLoc = 0 use the local input file (local.out)
= 1 use customized mesh
(mesh.fem, xt.in, xt.J1, rsdinput.tcl)

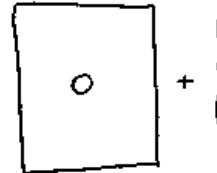
defaults:
anaReuse = 0
anaGenLoc = 0

2) ALTEPALT::Run -- run elastic-plastic analysis

ALTEPALT::Run anaReuse anaGenLoc

where anaReuse = 0 stiffness matrix can not be reused
= 1 can be reused
anaGenLoc = 0 use the local input file (local.out)
= 1 use customized mesh
(mesh.fem, xt.in, xt.J1, rsdinput.tcl)

defaults:
anaReuse = 0
anaGenLoc = 0



3) ALTRsd::Rsd -- run residual strength analysis

ALTRsd::Rsd anaType anaReuse anaGenLoc

where anaType = elastic elastic analysis
= plastic elastic-plastic analysis
anaReuse = 0 stiffness matrix can not be reused
= 1 can be reused
anaGenLoc = 0 use the local input file (local.out)
= 1 use customized mesh
(mesh.fem, xt.in, xt.J1, rsdinput.tcl)

defaults:
anaType = elastic
anaReuse = 0
anaGenLoc = 0

4) ALTear::App -- run table tearing analysis (application phase)
Note: analysis stops when MSD cracks are linked up with the lead crack.
Therefore, this can not be used to analyze the tearing of a single crack.

ALTear::App plotItem anaReuse anaGenLoc

```

where plotItem = {} nothing to be plotted
      = (other options are used to generate snap-shot of
          strain/stress contour when image-processing
          packages are installed (ImagMagick and
          ghostscript)
anaReuse = 0 stiffness matrix can not be reused
      = 1 can be reused
anaGenLoc = 0 use the local input file (local.out)
      = 1 use customized mesh
          (mesh.fem, xt.in, xt.J1, rsdinput.tcl)

defaults:
  plotItem = {}
  anaReuse = 0
  anaGenLoc = 0

```

-
- 5) Ploting commands (available in interactive mode. In scripting mode, some tricky is needed. The enviroment variable DISPLAY must set to direct the windows to an X-server. At the begin of the script, use the command "source ~/.tkaltrc" to initialize the graphics part. Then, these command becomes available.)

- (a) To prepare a window for plot, use the following command

```
source $ALTLibDir/ALTplot; mtvInit mtv ← mtvInit
```

- (b) To plot stress/strain or mesh, use the following

```
ALTPlot opt; .mtv.mtv -readMesh
```

```
where opt = m plot mesh
      = t1 sigma_{11}
      = t2 sigma_{12}
      = t3 sigma_{22}
      = e equivalent plastic strain
      = w stress work density
      = s1 epsilon_{11}
      = s2 epsilon_{12}
      = s3 epsilon_{22}
```

- (c) To plot a data set in a file,, use the following

```
mtvInit
.mtv.mtv -read fileName
```

where fileName is the name of the file. For example, out.dat..

- (d) use the key "PgUp" and "PgDn" to switch between different pages in the ploting window. (Note: direct the keystrike to that window, not at the command prompt.) The commands for this are

```
mtvInit
.mtv.mtv -prev
.mtv.mtv -next
```

- (e) to generate a postscript copy of the plot

```
mtvInit
.mtv.mtv -print fileName
where fileName is the file where the postscript will be written.
```

- (d) to close the window

```
destroy .mtv
```

A Brief description of some of the files for 2DFEAM code

Local analysis is a plane stress analysis of AL 2024T3. The input file can be a single file (local.out); or several other files (mesh.fem, xt.in, xt.J1, rsdinput.tcl). When the single file (local.out) is used, the automatic mesh generate will create the other input files based on this file.

Analysis type includes

- a) elastic alternating analysis
- b) elastic-plastic alternating analysis
- c) elastic residual strength analysis
- d) elastic-plastic residual strength analysis
- e) stable tearing analysis
- f) command driven analysis

in a) elastic FEA is used to compute the crack tip SIF
in b) EPFEAM is used to compute T^* (J) at the specified crack tip contours
in c) the residual strength of the panel is computed, assuming that the load level is proportional to all the specified BC.
in d) residual strength based on EPFEAM and T^* (J) (an equivalent KIC is specified)
in e) a stable tearing analysis. A lot more testing is needed. Special care is needed.
in f) the mesh and boundary condition is first read in. Then, drive the analysis using the TCL commands. Detailed knowledge is required. It stops when all the cracks are linked up, i.e. there is only one crack left.

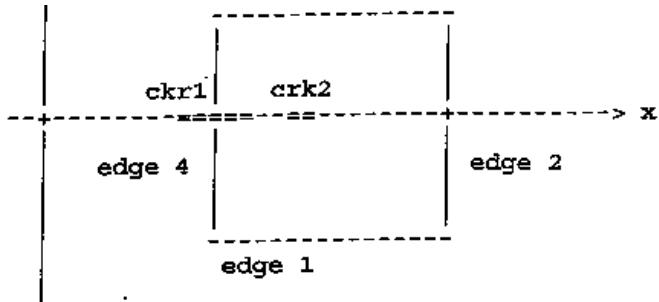
I. input file for local model (local.out)

The input file for local model contains the information for automatic generation of the mesh for the local model. The mesh generator considers only a rectangular local model, with/without rivet holes.

The first section of the input file specifies the informations about the cracks. First, no. of cracks and crack tip coordinates are specified. All cracks are on the x-axis. Therefore, only x-coordinates are specified. Then, a list of crack tip coordinates is specified. An edge crack is specified by letting one of the crack tip locate outside the body. Cracks at left hand side must appear earlier than those at the right hand side.

The second section specifies the boundary conditions at each edge. Boundary conditions at each edge is represented by values of a list of nodes. The traction/displacement in between the points are obtained using linear interpolation. First the number of nodes on each edge is specified. Then, the boundary conditions at each edge is specified. The four edges are indicated as the following.





The boundary condition type is
 code x-direction y-direction
 0 traction traction
 1 displacement displacement
 2 displacement traction
 3 traction displacement

A typical line of boundary condition point is

x y fx fy

where (x,y) are the coordinates of the point. F_x, f_y is the prescribed value. They may be traction or displacement, determined by the specified code (for the type of boundary condition) for the edge. For edge 1 and 3, the x-coordinates of the BC points must be in increasing order. For edge 2 and 4, the y-coordinates of the points must be in increasing order. These nodes may not be the same as nodes in the FEM mesh (created by the mesh generator).

The third section is about the rivet holes. It also contains the information for mesh generation. All the holes are of the same radius. A list of x-coordinates (in increasing order) is given as the possible x-coordinates for the center of rivet holes. Similarly, a list of y-coordinates is given for the possible locations for the rivet holes. These numbers can be introduced to the control the mesh generation. When there is no hole, the specified radius of the hole will control the mesh size. The center of holes is specified by the indices of the x/y coordinates in the list. A typical line for rivet hole is

nx ny Px Py

where (Px, Py) are the (x,y) components of the pin load. The nx 'th coordinate in the list for the x-coordinates is the x-coordinate of the hole. Similarly, ny is the index for the y-coordinate of the hole. The line with non-positive indices marks the end of the list of holes. The unit for the displacement is inch; and ksi for stress.

Finally, other informations are specified. Some parameters are listed below. They will be used to set the corresponding internal variable. There are essentially TCL commands.

1. the load level index for the currently specified B.C

set pressure *x*

where *x* is the load level index. Typically, it is the far field stress.

It is used in the residual strength/stable tearing analysis.

2. critical SIF

set KIC x

where x is the critial SIF in (Ksi in^{0.5})

3. set T* resistance curve

crack::config -tStar {0.1+sqrt(\$da)}

The formula must be inside {}. \$da is the key word. It stands for the amount of crack extension. If not specified, a defaut curve will be used.

4. set epsilon for the T* curve

crack::config -epsilon 0.05

If not specified, a defaut value will be used.

5. set the amount of crack extension for each crack growth step in stable tea

crack::config -crkIncr 0.04

If not specified, a defaut value will be used.

Example (local.out):

```
=====
4          # number of cracks
-10 10    # (left, right) crack-tip coordinates for the 1'th crack
10.5 11   # second crack
11.5 12   # third
12.5 13   # the last one
3 2 0 0   ##### boundary conditon types for edge 1,2,3,4
2 2 2 2   ##### no. of points used for specify BC for each edge
0 -20 0 0  # the first point on the first edge (x,y,fx,fy)
45 -20 0 0 # the second (last) point on the second edge
45 -20 0 0 # BC for the second edge      fnt
45 20 0 0  # for the second edge
0 20 0 1  # BC for the third
45 20 0 1  # for the third edge
0 -20 0 0  # BC
0 20 0 0  #
1 1 3     # no. of (x,y) coordinates and the radius of the hole 4 2 1.
20        # list of x-coordinates, seperated by space           0.1 0.2 0.
0        # list of y-coordinates                           0.1 0.2
-1 -1 0 0  # the end of list for rivet holes
set pressure 1; # other control parameters
set KIC 90;    # ...
crack::config -tStar {0.1+sqrt($da)}
crack::config -epsilon "0.05"
crack::config -crkIncr "0.04"
                                         X=0.1 - - -
                                         Y=0.2   0.5 0.5 0.5
                                         Y=0.1   0   0   0
                                         3   Px Py
```

II. input file for the FEM mesh (mesh.fem)

The code accepts currently only 8-node elements. The load is specified as distributed load at element edge. Current GUI does not take point loads.

The first section is the nodal list. A typical line is

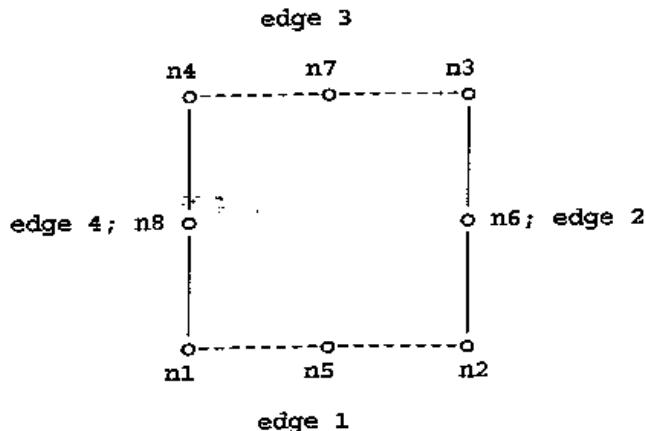
ID x y

where ID is the nodal number and (x,y) are coordinates. A negative ID marks the end of the nodal list

The second section is the element list. A typical line is

n1 n2 n3 n4 n5 n6 n7 n8

where n1-n8 are node IDs as specified in the first section. Negative IDs marks the end of the element list. The numbering scheme is shown in the following. The edges are also numbered as the following.



The third part is the nodal constraints. A typical line is

ID code fx fy

where ID is the nodal ID as in the first section. The code is

code	constraint-type
1	u is specified, $fx=u$, <mathfy< math=""> has no meaning</mathfy<>
2	v is specified, $fy=v$, fx has no meaning
3	both u,v are specified, $fx=u$, $fy=v$

A negative ID marks the end of the nodal constraints.

The last section is the edge loads. A typical line is

```
eleID edgeID fx1 fy1 fx2 fy2 fx3 fy3
```

where it indicates that the load is on the eleID'th element, the edgeID'th edge. fx1, fx2, fx3 are the values (Tx) at the three nodes of the edge. Tx is the x-component of the traction. Similarly, fy1, fy2, fy3 are the values of the y-component of the tractions. A negative eleID marks the end of the edge loads.

example (mesh.fem):

```
=====
1 0 0      # the nodal list
2 2 0
3 2 2
4 0 2
5 1 0
6 2 1
7 1 2
8 0 1
-1 0 0    # end of nodal list (three numbers are needed at this line)
1 2 3 4 5 6 7 8    # the element list
-1 0 0 0 0 0 0 0    # end of element list
1 3 0 0          # the nodal constraints (both u,v are fixed at node 1)
5 2 0 0
2 2 0 0
-1 0 0 0          # end of the nodal constraint list
1 3 0.1 0 0.1 0.5 0.1 1.0 # edge load list
-1 0 0 0 0 0 0 0    # end
=====
```

III. input file for the cracks (xt.in)

The first part is same as the first section in the input file for the local model (local.out)

Then, points where displacement in x/y direction is constrained for the analytical solution is specified. (Do not specify a point on the crack surface.) It requires two lines. The first line specifies the x,y coordinates of the point and the prescribed u. The second line specifies the x,y coordinates of the second point and the prescribed v. The two points can be at the same location.

Example (xt.in):

```
=====
4          # number of cracks
-10 10     # (left, right) crack-tip coordinates for the 1'th crack
10.5 11    # second crack
11.5 12    # third
12.5 13    # the last one
20 0 0      # u is 0 for the analytical solution at (20,0)
=====
```

```
20 0 0      # v is 0 for the analytical solution at (20,0)
```

```
IV. input file for the contours (xt.J1)
```

The first line *contains* the number of contours in the file. A contour is specified by a list of nodes. Contours are separated by single line. These nodes can not be located exactly on the crack surface.

```
Example (xt.J1):
```

```
2          # two contours
 0.0 -0.0001  # the first node at (0,0)
15 0.0 -1.0   # the second node at (0,-1.0). use 15 points between the
25 1.0 -1.0   # the first and second node. use 25 points between the
30 1.0 1.0    # the second and third node.
25 0.0 1.0
15 0.0 0.0001
0 0 0        # this marks the end of the contour

 0.0 -0.0001  # the first node at (0,0)
15 0.0 -1.5   # the second node at (0,-1.5). use 15 points between the
25 1.5 -1.5   # the first and second node. use 25 points between the
30 1.5 1.5    # the second and third node.
25 0.0 1.5
15 0.0 0.0001
0 0 0        # this marks the end of the contour
```

```
V. file for other inputs (rsdinput.tcl)
```

This part is almost the same as the last section in local.out. The variable "crks" must be set explicitly in rsdinput.tcl when local.out is not used.

```
example (rsdinput.tcl)
```

```
set crks {
 {-10.000000 10.000000}
 {10.500000 11.000000}
 {11.500000 12.000000}
 {12.500000 13.000000}
 };
set ALTCrkBodySize {0.000000 45.000000}; # the xmin and xmax
set pressure 1;                            # the load level index
set KIC 90;                                # critical (equivalent) SIF
```

The variable crks contains the list of cracks.

The variable ALTCrkBodySize contains the min and max x-coordinates for the body at y=0 (where the crack line is). It is used to determine if there is an edge crack.

VI. output file 1 (out.dat)

Some major results are put in out.dat by the GUI.

VII. output file 2 (xt.mtv)

Right after a contour plot for the stress, or the plastic strain, xt.mtv contains corresponding nodal values. It is listed for each element at each node. The nodal coordinates followed by corresponding value. The stress is the total elasto-plastic stress. The plastic strain is the equivalent plastic strain.